ELSEVIER

The Journal of Logic and Algebraic Programming 47 (2001) 15–45

THE JOURNAL OF LOGIC AND ALGEBRAIC PROGRAMMING

www.elsevier.com/locate/jlap

# Variable binding operators in transition system specifications

C.A. Middelburg[a,b]

[a] *Computing Science Department, Eindhoven University of Technology, P.O. Box 513, 5600 MB Eindhoven, Netherlands*

[b] *Department of Philosophy, Utrecht University, P.O. Box 80126, 3508 TC Utrecht, Netherlands*

Received 22 March 2000; received in revised form 17 August 2000; accepted 17 August 2000

**Abstract**

In this paper the approach to structural operational semantics (SOS) using transition system specifications (TSSs) is extended to deal with variable binding operators and many-sortedness. Bisimulation and Verhoef's transition rule format, known as the panth format, generalize naturally to the new TSSs. It is shown that in this setting bisimulation is still a congruence for meaningful TSSs in panth format. Formats guaranteeing that bisimulation is a congruence are important for the application of TSSs to provide process calculi, and programming and specification languages, with an operational semantics. The new congruence result is relevant because in many of these applications, variable binding operators and many-sortedness are involved. It is also sketched how the presented approach can be further extended to deal with given sorts and parametrized transition relations. Given sorts are useful if the semantics of the terms of certain sorts has been given beforehand. This happens frequently in practice, as does the often related need of parametrized transition relations. 

*Keywords:* Structural operational semantics; Transition system specifications; Panth format; Bisimulation; Congruence; Variable binding operators; Many-sortedness; Binding terms; Binding algebras; Partial stable models

## 1. Introduction

In Ref. [24], an approach to structural operational semantics (SOS) using transition system specifications (TSSs) was introduced. The approach considers transition systems where the states are the closed terms over a given signature. The original TSSs define binary transition relations by means of transition rules with positive premises. The approach has first been extended in Refs. [10,22] to transition rules

*E-mail address:* keesm@win.tue.nl (C.A. Middelburg).

1567-8326/01/$ - see front matter 
PII: S1567-8326(00)00003-5

with positive and negative premises and next in Ref. [38] to the specification of unary and binary relations. In all these cases, variable binding operators and many-sortedness are not supported. In many applications of TSSs, it is convenient to use negative premises or to define unary relations (see, e.g., Refs. [22,38]). Negative premises can often be avoided, but at the expense of simplicity. Representation of unary relations by binary relations is possible, but this trick does not contribute to comprehensibility.

In many applications of TSSs, it is in addition necessary to have support for many-sortedness or variable binding operators. Many-sortedness is found, for example, in process algebras with timing (see, e.g., Refs. [4,11,30]). Examples of variable binding operators are the integration operator $\int$ of real time ACP [4] and the recursion operator $\mu$ of CSP [25] and CCS [28]. Using transition rules to cope with many-sortedness is unpractical and obscures the fact that it is a static matter. Variable binding operators cannot be coped with at all without further extension. In Ref. [17], an extension to deal with variable binding operators and many-sortedness is proposed. Another one is proposed in this paper. An important difference between the extension presented in Ref. [17] and the one presented in this paper is that in the latter distinction between formal and actual variables, formal and actual terms, formal and actual substitutions, etc. is not made. This leads to TSSs that are more closely related to the original ones than the TSSs of Ref. [17]. As a consequence, the transition rule format introduced in Ref. [38], known as the panth format, generalizes naturally to the new TSSs.

The main difference with the original TSSs is that terms are used in which operators may bind variables in their arguments and variables may have arguments. The terms concerned are essentially the binding terms investigated in Ref. [36]. Similar expressions were also part of the meta-language used in Ref. [2] to introduce Frege structures. Variables that may have arguments are also known from combinatory reduction systems [27], where they occur as meta-variables in schematic rewrite rules for term rewriting with bound variables. Having variables that may have arguments obviates the need to distinguish two kinds of variables, terms, substitutions, etc. in TSSs. Such a distinction, which is made in Ref. [17], hinders generalization of definitions and results concerning TSSs without support for variable binding operators. Besides, it is doubtful that the complexity introduced by the distinction pays off in terms of the extent of applicability.

The meaning of TSSs proposed in Ref. [10], and reformulated in Ref. [21], also generalizes naturally to the TSSs presented in this paper. In Ref. [10], the meaning of a TSS is defined in a way that facilitates proving certain theorems related to the use of stratification (see, e.g., Ref. [22]) as a technique to check if a TSS is meaningful. In Ref. [21], it is defined in a way that makes comparison with potential alternatives easier. In this paper, the meaning of TSSs is defined in still another way aimed at clarity in this intricate issue. It goes without saying that the different definitions agree with each other.

Support for many-sortedness has some interesting consequences. It happens frequently in practice that the semantics of the terms of certain sorts, called given sorts, has been given beforehand. The sort that represents the time domain in process algebras with timing is a typical example. It is impractical and unnecessary to redefine the semantics of the terms of given sorts. Furthermore, distinguishing given sorts makes it possible to relax the panth format and to deal

with transition relations parametrized by terms of given sorts. This is also discussed in this paper.

The structure of this paper is as follows. First of all, in Section 2, we present binding terms and TSSs that define transition relations on binding terms. Binding algebras, the structures in which binding terms are interpreted, are presented in Section 2 as well. In Section 3, we discuss the meaning of TSSs. Then, in Section 4, we first define the bisimulation equivalence induced by TSSs and the panth format for TSSs, and then show that for meaningful TSSs in panth format bisimulation equivalence is a congruence. After that, in Section 5, we explain how to deal with given sorts and parametrized transition relations using the TSSs introduced in Section 2. Finally, in Section 6, we make some concluding remarks.

## 2. Binding terms and TSSs

In this section, we first introduce the notions of binding term and binding algebra. The latter are the structures in which binding terms are interpreted. Next, we generalize the notion of TSS from conventional terms to binding terms. The meaning of TSSs is discussed in Section 3.

Abstract notions of binding term and binding algebra were introduced in Ref. [14]. The kind of binding terms and binding algebras introduced in this section are essentially many-sorted versions of the ones that were first introduced in Ref. [36].

### 2.1. Binding terms

We define terms over a many-sorted signature, roughly speaking a collection of sorts and operators, and a variable domain. Therefore, we first define the notions of binding sort, many-sorted binding signature and variable domain.

We assume a set $\mathscr{S}$ of base *sorts*. A base sort stands for a set of which the elements are called ordinary objects. A binding sort is either a base sort or a sort that stands for a set of functions from certain sets of ordinary objects to a certain set of ordinary objects – all determined by given base sorts. As explained below, sorts of the latter kind are used for variable binding in arguments of operators.

**Definition 2.1.** Let $S \subseteq \mathscr{S}$. Then the set $\mathscr{B}(S)$ of *binding sorts* over $S$ is inductively defined by the following formation rules ($n > 0$):
1. $S \subseteq \mathscr{B}(S)$;
2. $s_1, \ldots, s_n,\ s \in S \Rightarrow s_1, \ldots, s_n \,.\, s \in \mathscr{B}(S)$.

We assume a $\mathscr{B}(\mathscr{S})^* \times \mathscr{S}$-indexed family of mutually disjoint sets of *binding operators* $\mathscr{O} = \langle \mathscr{O}_\tau \rangle_{\tau \in \mathscr{B}(\mathscr{S})^* \times \mathscr{S}}$. We also assume a $\mathscr{B}(\mathscr{S})$-indexed family of mutually disjoint, countably infinite sets of *variables* $\mathscr{V} = \langle \mathscr{V}_\mathfrak{s} \rangle_{\mathfrak{s} \in \mathscr{B}(\mathscr{S})}$. It is assumed that the sets $\bigcup_{\tau \in \mathscr{B}(\mathscr{S})^* \times \mathscr{S}} \mathscr{O}_\tau$ and $\bigcup_{\mathfrak{s} \in \mathscr{B}(\mathscr{S})} \mathscr{V}_\mathfrak{s}$ are disjoint. We write $o : \mathfrak{s}_1 \times \cdots \times \mathfrak{s}_n \to s$ to indicate that $o \in \mathscr{O}_{(\langle \mathfrak{s}_1, \ldots, \mathfrak{s}_n \rangle, s)}$ and we write $x : \mathfrak{s}$ to indicate that $x \in \mathscr{V}_\mathfrak{s}$. A variable $x$ is called *ordinary* if $x : s$ for some $s \in \mathscr{S}$.

An operator $o : \mathfrak{s}_1 \times \cdots \times \mathfrak{s}_n \to s$ has $n$ arguments. If $\mathfrak{s}_i = s_{i1}, \ldots, s_{in_i} \,.\, s_i$ $(1 \leqslant i \leqslant n)$, then it binds $n_i$ variables, of base sorts $s_{i1}, \ldots, s_{in_i}$, in the $i$th argument. Otherwise, i.e., if $\mathfrak{s}_i \in S$, it does not bind any variable in the $i$th argument.

**Definition 2.2.** A (*many-sorted*) *binding signature* is a pair $\Sigma = (S, O)$, with $S \subseteq \mathscr{S}$ and $O \subseteq \mathscr{O}$, such that for all $o \in O$, if $o : \mathfrak{s}_1 \times \cdots \times \mathfrak{s}_n \to s$, then $\mathfrak{s}_1, \ldots, \mathfrak{s}_n, s \in \mathscr{B}(S)$.

**Definition 2.3.** Let $\Sigma = (S, O)$ be a binding signature. Then a *variable domain* for signature $\Sigma$ is a set $X \subseteq \bigcup_{\mathfrak{s} \in \mathscr{B}(S)} \mathscr{V}_{\mathfrak{s}}$ such that for all $s \in S$, $\mathscr{V}_s \subseteq X$. For every $\mathfrak{s} \in \mathscr{B}(S)$, we write $X_{\mathfrak{s}}$ for the set $\{x \in X \mid x : \mathfrak{s}\}$. We write $\mathrm{OV}(X)$ and $\mathrm{PV}(X)$ for the sets $\{x \in X \mid x \text{ is ordinary}\}$ and $\{x \in X \mid x \text{ is not ordinary}\}$, respectively. We write $\mathscr{X}_\Sigma$ and $\mathscr{V}_\Sigma$ for the variable domains $\bigcup_{s \in S} \mathscr{V}_s$ and $\bigcup_{\mathfrak{s} \in \mathscr{B}(S)} \mathscr{V}_{\mathfrak{s}}$, respectively.

Next, in Definition 2.4, we define the notion of binding term. Formation rule 2 shows that variables, with the exception of ordinary variables, have arguments. Variables are bound in terms formed by formation rule 3. Notice that the terms formed by application of this rule serve only as arguments of operators.

**Definition 2.4.** Let $\Sigma = (S, O)$ be a binding signature and $X$ be a variable domain for $\Sigma$. Then $\mathscr{T}_\Sigma(X) = \langle \mathscr{T}_\Sigma(X)_{\mathfrak{s}} \rangle_{\mathfrak{s} \in \mathscr{B}(S)}$, the family of sets of *binding terms* over signature $\Sigma$ and variables $X$, is inductively defined by the following formation rules:
1. $x \in \mathrm{OV}(X)$, $x : s \Rightarrow x \in \mathscr{T}_\Sigma(X)_s$;
2. $x \in \mathrm{PV}(X)$, $x : s_1, \ldots, s_n \,.\, s$, $t_1 \in \mathscr{T}_\Sigma(X)_{s_1}, \ldots, t_n \in \mathscr{T}_\Sigma(X)_{s_n} \Rightarrow x(t_1, \ldots, t_n) \in \mathscr{T}_\Sigma(X)_s$;
3. $x_1, \ldots, x_n \in \mathrm{OV}(X)$ and mutually distinct, $x_1 : s_1, \ldots, x_n : s_n$, $t \in \mathscr{T}_\Sigma(X)_s \Rightarrow x_1, \ldots, x_n \,.\, t \in \mathscr{T}_\Sigma(X)_{s_1, \ldots, s_n \,.\, s}$;
4. $o \in O$, $o : \mathfrak{s}_1 \times \cdots \times \mathfrak{s}_n \to s$, $t_1 \in \mathscr{T}_\Sigma(X)_{\mathfrak{s}_1}, \ldots, t_n \in \mathscr{T}_\Sigma(X)_{\mathfrak{s}_n} \Rightarrow o(t_1, \ldots, t_n) \in \mathscr{T}_\Sigma(X)_s$.

In $o(t_1, \ldots, t_n)$, we usually omit the parentheses whenever $n = 0$. We write $\mathscr{T}_\Sigma$ for $\mathscr{T}_\Sigma(\mathscr{V}_\Sigma)$. For $t \in \bigcup_{\mathfrak{s} \in \mathscr{B}(S)} \mathscr{T}_\Sigma(X)_{\mathfrak{s}}$, we write $\mathrm{s}(t)$ for the $\mathfrak{s} \in \mathscr{B}(S)$ such that $t \in \mathscr{T}_\Sigma(X)_{\mathfrak{s}}$.

**Example 2.5.** In CCS [28], the operator $\mu$ is used to define processes recursively. For example, the expression $\mu x \,.\, ax$ denotes the solution of the equation $x = ax$, i.e., the process that will keep on performing action $a$ forever. The recursion operator $\mu$ is actually a unary variable binding operator that binds one variable in its argument. The expression $\mu x \,.\, ax$ abbreviates the binding term $\mu(x \,.\, ax)$.

The following definition makes the notion of closed binding term precise. It also introduces the notions of free and bound occurrence of a variable.

**Definition 2.6.** An occurrence of a variable $x$ in a term $t$ is *bound* if the occurrence is in the term $t'$ of a subterm of the form $x_1, \ldots, x_n \,.\, t'$ where $x \in \{x_1, \ldots, x_n\}$; otherwise it is *free*. If $x$ has at least one free occurrence in $t$, it is called a *free variable* of $t$. A term $t$ is *closed* if it is a term without free variables. We write $\mathscr{CT}_\Sigma(X)$ for the family of sets of closed binding terms $\langle \{t \in \mathscr{T}_\Sigma(X)_{\mathfrak{s}} \mid t \text{ is closed}\} \rangle_{\mathfrak{s} \in \mathscr{B}(S)}$. We write $\mathscr{CT}_\Sigma$ for $\mathscr{CT}_\Sigma(\mathscr{V}_\Sigma)$.

Substitution of binding terms for variables is needed in many occasions. We first define a notion of substitution restricted to ordinary variables. It allows us to define the notion of term algebra in the setting of binding algebras (see Section 2.2). Using the notion of term algebra, we will define a notion of substitution that is not restricted to ordinary variables. Notice that free and bound occurrences of variables are treated differently in substitution.

**Definition 2.7.** Let $\Sigma = (S, O)$ be a binding signature and $X$ be a variable domain for $\Sigma$. Then an *ordinary substitution* $\sigma : X \to \mathscr{T}_\Sigma(X)$ of terms in $\mathscr{T}_\Sigma(X)$ for ordinary variables in $X$ is an $S$-indexed family of functions $\langle \sigma_s : X_s \to \mathscr{T}_\Sigma(X)_s \rangle_{s \in S}$. An ordinary substitution $\sigma$ extends from ordinary variables to terms in the usual way: $\sigma_{\mathfrak{s}(t)}(t)$ is the term obtained by simultaneously replacing in $t$ all free occurrences of ordinary variables $x$ by $\sigma_{\mathfrak{s}(x)}(x)$, renaming bound occurrences of ordinary variables in $t$ if needed to avoid free occurrences of variables in the replacing terms becoming bound. For every ordinary substitution $\sigma : X \to \mathscr{T}_\Sigma(X)$ and $t \in \mathscr{T}_\Sigma(X)_s$, we write $\sigma(t)$ or $t\sigma$ for $\sigma_s(t)$. We write $[t_1, \ldots, t_n/x_1, \ldots, x_n]$ for the ordinary substitution $\sigma$ such that $\sigma(x_1) = t_1, \ldots, \sigma(x_n) = t_n$ and $\sigma(x) = x$ if $x \notin \{x_1, \ldots, x_n\}$.

Notice that ordinary substitution is defined up to change of bound variables. This does not pose any problem, because binding terms that can be obtained from each other by change of bound variables are semantically equivalent (see Definition 2.9). Indeed, we will introduce an equivalence relation formalizing this identification (see Definition 2.11).

### 2.2. Binding algebras

Binding terms are interpreted in binding algebras. Binding algebras constitute a restricted kind of second-order algebras, suitable to deal with variable binding operators, which can be regarded as an algebraic generalization of the Frege structures introduced in Ref. [2].

We define binding algebras with respect to a binding signature. An important condition to be satisfied by a binding algebra is that each term can be given an interpretation in it for any object or function its free variables may stand for. Obviously, the interpretation of a term depends on the objects and functions that are associated with its free variables. Assignments, which are defined first, model such associations. They can be viewed as semantic counterparts of substitutions that are not restricted to ordinary variables (see Definition 2.16).

**Definition 2.8.** Let $\Sigma = (S, O)$ be a binding signature and $X$ be a variable domain for $\Sigma$. Furthermore, let $\mathscr{D}$ be a $\mathscr{B}(S)$-indexed family of sets $\langle \mathscr{D}_{\mathfrak{s}} \rangle_{\mathfrak{s} \in \mathscr{B}(S)}$. Then an *assignment* $\alpha : X \to \mathscr{D}$ of values in $\mathscr{D}$ to variables in $X$ is a $\mathscr{B}(S)$-indexed family of functions $\langle \alpha_{\mathfrak{s}} : X_{\mathfrak{s}} \to \mathscr{D}_{\mathfrak{s}} \rangle_{\mathfrak{s} \in \mathscr{B}(S)}$. We write $[X \to \mathscr{D}]$ for the set of all assignments $\alpha : X \to \mathscr{D}$. For every assignment $\alpha : X \to \mathscr{D}$ and $x : \mathfrak{s}$, we write $\alpha(x)$ for $\alpha_{\mathfrak{s}}(x)$. For every assignment $\alpha : X \to \mathscr{D}$, $x : \mathfrak{s}$ and $d \in \mathscr{D}_{\mathfrak{s}}$, we write $\alpha(x \to d)$ for the $\alpha' : X \to \mathscr{D}$ such that $\alpha'(y) = \alpha(x)$ if $y \neq x$ and $\alpha'(x) = d$.

In Definition 2.9, binding algebras are defined. Rules 1–3 make precise what the intended interpretation of binding sorts and binding operators are. Rules 1 and 3 are familiar from ordinary many-sorted algebras. Rule 2 shows that binding sorts other than base sorts are interpreted as sets of functions. Rule 4 makes the above-mentioned condition on the interpretability of terms in binding algebras precise. In the terminology of Ref. [2], this condition is equivalent to an explicit closure condition for the interpretations of the binding sorts, and an $\mathscr{F}$-functional condition for the interpretation of each binding operator (here $\mathscr{F} = \langle \mathscr{D}_{\mathfrak{s}} \rangle_{\mathfrak{s} \in \mathscr{B}(S)}$).

**Definition 2.9.** Let $\Sigma = (S, O)$ be a binding signature. Then a *binding algebra* with signature $\Sigma$ is a pair $\mathscr{A} = (\langle \mathscr{D}_{\mathfrak{s}} \rangle_{\mathfrak{s} \in \mathscr{B}(S)}, \langle \mathscr{I}_o \rangle_{o \in O})$, where

1. for each $\mathfrak{s} \in \mathscr{B}(S)$, $\mathscr{D}_{\mathfrak{s}}$ is a non-empty set, called the *carrier set* for $\mathfrak{s}$;
2. for each $\mathfrak{s} \in \mathscr{B}(S) \setminus S$, $\mathfrak{s} = s_1, \ldots, s_n \,.\, s$, $\mathscr{D}_{\mathfrak{s}} \subseteq \mathscr{D}_{s_1} \times \cdots \times \mathscr{D}_{s_n} \to \mathscr{D}_s$;
3. for each $o \in O$, $o : \mathfrak{s}_1 \times \cdots \times \mathfrak{s}_n \to s$, $\mathscr{I}_o$ is a function $\mathscr{I}_o : \mathscr{D}_{\mathfrak{s}_1} \times \cdots \times \mathscr{D}_{\mathfrak{s}_n} \to \mathscr{D}_s$, called the *denotation* of $o$;
4. there exists a family of functions

$$[\![\_]\!]\_ = \langle [\![\_]\!]\__{\mathfrak{s}} : \mathscr{T}_{\Sigma_{\mathfrak{s}}} \times [\mathscr{V}_{\Sigma} \to \langle \mathscr{D}_{\mathfrak{s}} \rangle_{\mathfrak{s} \in \mathscr{B}(S)}] \to \mathscr{D}_{\mathfrak{s}} \rangle_{\mathfrak{s} \in \mathscr{B}(S)}$$

such that for all terms and assignments (writing $[\![t]\!]_{\alpha}$ for $[\![t]\!]_{\alpha_{\mathfrak{s}(t)}}$):

(a) $[\![x]\!]_{\alpha} = \alpha(x)$;
(b) $[\![x(t_1, \ldots, t_n)]\!]_{\alpha} = \alpha(x)([\![t_1]\!]_{\alpha}, \ldots, [\![t_n]\!]_{\alpha})$;
(c) $[\![x_1, \ldots, x_n \,.\, t]\!]_{\alpha}$ is the function $f \in \mathscr{D}_{\mathfrak{s}(x_1),\ldots,\mathfrak{s}(x_n) \,.\, \mathfrak{s}(t)}$ such that for all $d_1 \in \mathscr{D}_{\mathfrak{s}(x_1)}, \ldots, d_n \in \mathscr{D}_{\mathfrak{s}(x_n)}$: $f(d_1, \ldots, d_n) = [\![t]\!]_{\alpha(x_1 \to d_1) \ldots (x_n \to d_n)}$;
(d) $[\![o(t_1, \ldots, t_n)]\!]_{\alpha} = \mathscr{I}_o([\![t_1]\!]_{\alpha}, \ldots, [\![t_n]\!]_{\alpha})$.

For a given binding algebra $\mathscr{A} = (\mathscr{D}, \mathscr{I})$, $[\![\_]\!]\_$ is uniquely determined and is called the family of *evaluation* functions associated with $\mathscr{A}$. Furthermore, an assignment $\alpha : X \to \mathscr{D}$, where $X$ is a variable domain for $\Sigma$, is called an assignment *in* $\mathscr{A}$.

Notice that a term of the form $x$ or $o(t_1, \ldots, t_n)$ is evaluated as in the case of ordinary many-sorted algebras. Notice further that a term of the form $x_1, \ldots, x_n \,.\, t$ can only be evaluated if it denotes a function that is an element of the carrier set for the sort of the term.

**Example 2.10.** We consider the recursion operator from Example 2.5 again. Let $\Sigma$ be a binding signature that includes, among other things, a sort $\mathsf{P}$ of processes, a constant $0 :\to \mathsf{P}$, unary operators $a : \mathsf{P} \to \mathsf{P}$ (for certain actions $a$), and a unary operator $\mu : \mathsf{P} \,.\, \mathsf{P} \to \mathsf{P}$. Let $\mathscr{A} = (\mathscr{D}, \mathscr{I})$ be a binding algebra with signature $\Sigma$. The denotation $\mathscr{I}_{\mu}$ of $\mu$ is a function $\mathscr{I}_{\mu} : (\mathscr{D}_{\mathsf{P}} \to \mathscr{D}_{\mathsf{P}}) \to \mathscr{D}_{\mathsf{P}}$. Notice that, in order to be a binding algebra with signature $\Sigma$, it is not required that $f(\mathscr{I}_{\mu}(f)) = \mathscr{I}_{\mu}(f)$ for all $f : \mathscr{D}_{\mathsf{P}} \to \mathscr{D}_{\mathsf{P}}$, i.e., application of $\mathscr{I}_{\mu}$ does not have to yield a fixed point. The binding term $\mu x \,.\, ax$ is evaluated in $\mathscr{A}$ as follows: $[\![\mu x \,.\, ax]\!]_{\alpha} = \mathscr{I}_{\mu}(\mathscr{I}_a)$. If $\alpha$ assigns $\mathscr{I}_a$ to $z$, i.e., $\alpha(z) = \mathscr{I}_a$, the binding term $z(\mu x \,.\, ax)$ is evaluated as follows: $[\![z(\mu x \,.\, ax)]\!]_{\alpha} = \mathscr{I}_a(\mathscr{I}_{\mu}(\mathscr{I}_a)) (= [\![a\mu x \,.\, ax]\!]_{\alpha})$.

Interesting among the binding algebras for a given signature are the term algebras. For their construction, we need an equivalence relation on the binding terms over the binding signature concerned. Therefore, we first define this equivalence

relation. It is a version of α-conversion. It identifies terms that can be obtained from each other by a change of bound variables. Notice that this equivalence relation is a congruence by construction.

**Definition 2.11.** Let $\Sigma = (S, O)$ be a binding signature and $X$ be a variable domain for $\Sigma$. Then $\dot{\approx}$ is the $\mathscr{B}(S)$-indexed family of least equivalence relations $\langle \dot{\approx}_{\mathfrak{s}} \subseteq \mathscr{T}_\Sigma(X)_{\mathfrak{s}} \times \mathscr{T}_\Sigma(X)_{\mathfrak{s}} \rangle_{\mathfrak{s} \in \mathscr{B}(S)}$ such that, writing $t \dot{\approx} t'$ for $t \dot{\approx}_{\mathrm{s}(t)} t'$:

1. $x_1, \ldots, x_n \,.\, t \dot{\approx} y_1, \ldots, y_n \,.\, t[y_1, \ldots, y_n / x_1, \ldots, x_n]$ (for mutually distinct variables $y_1, \ldots, y_n$ not free in $t$);
2. $x_1, \ldots, x_n \,.\, t \dot{\approx} y_1, \ldots, y_n \,.\, t' \Rightarrow \; t[z_1, \ldots, z_n / x_1, \ldots, x_n] \dot{\approx} t'[z_1, \ldots, z_n / y_1, \ldots, y_n]$ (for mutually distinct variables $z_1, \ldots, z_n$ not free in $x_1, \ldots, x_n \,.\, t$ or $y_1, \ldots, y_n \,.\, t'$);
3. $t \dot{\approx} t'$ and $x_1, \ldots, x_n \in \mathrm{OV}(X) \Rightarrow x_1, \ldots, x_n \,.\, t \dot{\approx} x_1, \ldots, x_n \,.\, t'$;
4. $t_1 \dot{\approx} t'_1, \ldots, t_n \dot{\approx} t'_n, x : \mathrm{s}(t_1), \ldots, \mathrm{s}(t_n) \,.\, s \Rightarrow x(t_1, \ldots, t_n) \dot{\approx} x(t'_1, \ldots, t'_n)$;
5. $t_1 \dot{\approx} t'_1, \ldots, t_n \dot{\approx} t'_n, o : \mathrm{s}(t_1) \times \cdots \times \mathrm{s}(t_n) \to s \Rightarrow o(t_1, \ldots, t_n) \dot{\approx} o(t'_1, \ldots, t'_n)$.

Furthermore, let $t \in \mathscr{T}_\Sigma(X)_{\mathfrak{s}}$ and $T \subseteq \mathscr{T}_\Sigma(X)_{\mathfrak{s}}$ for some $\mathfrak{s} \in \mathscr{B}(S)$. Then we write $[t]$ for $\{t' \in \mathscr{T}_\Sigma(X)_{\mathfrak{s}} \mid t \dot{\approx} t'\}$, and $[T]$ for $\{[t] \mid t \in T\}$.

It is easy to see that, for all $\mathfrak{s} \in \mathscr{B}(S)$ and $t, t' \in \mathscr{T}_\Sigma(X)_{\mathfrak{s}}$, $t \dot{\approx} t'$ implies that in all binding algebras $\mathscr{A} = (\mathscr{D}, \mathscr{I})$ with signature $\Sigma$ we have $[\![t]\!]_\alpha = [\![t']\!]_\alpha$ for all assignments $\alpha : X \to \mathscr{D}$.

**Example 2.12.** For the binding term $\mu x \,.\, ax$ used in Examples 2.5 and 2.10, we have $\mu x \,.\, ax \dot{\approx} \mu y \,.\, ay$ for variables $y : \mathsf{P}$.

To refer to the interpretations of binding terms in (binding) term algebras, we also introduce the notation $\bullet_{[t]}$. The intended meaning of $\bullet_{[t]}$ is simply $[t]$ if $t$ is not of the form $x_1, \ldots, x_n \,.\, t'$. The intended meaning of $\bullet_{[x_1, \ldots, x_n \,.\, t']}$ is a function on sets of $\dot{\approx}$-equivalence classes of terms, viz., the function of which application corresponds to taking the $\dot{\approx}$-equivalence class of the term obtained by substitution of representatives of the arguments concerned for $x_1, \ldots, x_n$ in $t'$.

**Definition 2.13.** Let $\Sigma = (S, O)$ be a binding signature and $X$ be a variable domain for $\Sigma$. Furthermore, let $t \in \mathscr{T}_\Sigma(X)_s$ for some $s \in S$ and let $x_1, \ldots, x_n \,.\, t' \in \mathscr{T}_\Sigma(X)_{\mathfrak{s}}$ for some $\mathfrak{s} \in \mathscr{B}(S) \setminus S$. Then we write $\bullet_{[t]}$ for $[t]$, and $\bullet_{[x_1, \ldots, x_n \,.\, t']}$ for the function $f : [\mathscr{T}_\Sigma(X)_{\mathrm{s}(x_1)}] \times \cdots \times [\mathscr{T}_\Sigma(X)_{\mathrm{s}(x_n)}] \to [\mathscr{T}_\Sigma(X)_{\mathrm{s}(t)}]$ such that for all $t_1 \in \mathscr{T}_\Sigma(X)_{\mathrm{s}(x_1)}$, $\ldots, t_n \in \mathscr{T}_\Sigma(X)_{\mathrm{s}(x_n)}$ we have $f([t_1], \ldots, [t_n]) = [t'[t_1, \ldots, t_n / x_1, \ldots, x_n]]$.

Notice that the function denoted by $\bullet_{[x_1, \ldots, x_n \,.\, t']}$ is well-defined because $\dot{\approx}$ is a congruence.

**Example 2.14.** Consider the binding signature $\Sigma$ used in Example 2.10 and suppose that $X$ is a variable domain for $\Sigma$. Let $x \in X$ be such that $x : \mathsf{P}$. Then $\bullet_{[x \,.\, ax]}$ denotes the function $f : [\mathscr{T}_\Sigma(X)_{\mathsf{P}}] \to [\mathscr{T}_\Sigma(X)_{\mathsf{P}}]$ such that for all terms $t \in \mathscr{T}_\Sigma(X)_{\mathsf{P}}$ we have $f([t]) = [at]$.

In Definition 2.15, term algebras are defined. Base sorts are interpreted as sets of $\dot{\approx}$-equivalence classes of terms. Binding sorts other than base sorts are interpreted as

sets of functions of the application-by-substitution kind described above on these sets of $\dot{\approx}$-equivalence classes of terms. Binding operators are interpreted as functions on these sets of either $\dot{\approx}$-equivalence classes or functions.

**Definition 2.15.** Let $\Sigma = (S, O)$ be a binding signature and $X$ be a variable domain for $\Sigma$. Then the *binding algebra of terms* with signature $\Sigma$ on $X$ is the binding algebra $\mathscr{A} = (\langle \mathscr{D}_\mathfrak{s} \rangle_{\mathfrak{s} \in \mathscr{B}(S)}, \langle \mathscr{I}_o \rangle_{o \in O})$, where
1. for each $\mathfrak{s} \in \mathscr{B}(S)$, $\mathscr{D}_\mathfrak{s} = \{\bullet_{[t]} \mid t \in \mathscr{T}_\Sigma(X)_\mathfrak{s}\}$;
2. for each $o \in O$, $o : \mathfrak{s}_1 \times \cdots \times \mathfrak{s}_n \to s$, $\mathscr{I}_o$ is the function $F : \mathscr{D}_{\mathfrak{s}_1} \times \cdots \times \mathscr{D}_{\mathfrak{s}_n} \to \mathscr{D}_s$ such that for all $t_1 \in \mathscr{T}_\Sigma(X)_{\mathfrak{s}_1}, \ldots, t_n \in \mathscr{T}_\Sigma(X)_{\mathfrak{s}_n}$ we have $F(\bullet_{[t_1]}, \ldots, \bullet_{[t_n]}) = \bullet_{[o(t_1,\ldots,t_n)]}$.

The binding algebra of terms with signature $\Sigma$ on $X$ is the free algebra with signature $\Sigma$ on $X$. It is the initial algebra with signature $\Sigma$ if $X = \mathscr{X}_\Sigma$.

The following definition shows that substitutions of binding terms over $\Sigma$ and $X$ for variables in $X$ are closely related to assignments in the binding algebra of terms with signature $\Sigma$ on $X$.

**Definition 2.16.** Let $\Sigma = (S, O)$ be a binding signature and $X$ be a variable domain for $\Sigma$. Furthermore, let $[\![\_]\!]_\_$ be the family of evaluation functions associated with the binding algebra of terms with signature $\Sigma$ on $X$. Then a *substitution* $\sigma : X \to \mathscr{T}_\Sigma(X)$ of terms in $\mathscr{T}_\Sigma(X)$ for variables in $X$ is a $\mathscr{B}(S)$-indexed family of functions $\langle \sigma_\mathfrak{s} : X_\mathfrak{s} \to \mathscr{T}_\Sigma(X)_\mathfrak{s} \rangle_{\mathfrak{s} \in \mathscr{B}(S)}$. The extension of $\sigma$ from variables to terms is a $\mathscr{B}(S)$-indexed family of functions $\langle \sigma_\mathfrak{s} : \mathscr{T}_\Sigma(X)_\mathfrak{s} \to \mathscr{T}_\Sigma(X)_\mathfrak{s} \rangle_{\mathfrak{s} \in \mathscr{B}(S)}$ such that for all $\mathfrak{s} \in \mathscr{B}(S)$ and $t \in \mathscr{T}_\Sigma(X)_\mathfrak{s}$ we have $[\sigma_\mathfrak{s}(t)] = [\![t]\!]_{\alpha_\mathfrak{s}}$, where $\alpha$ is the assignment $\alpha : X \to \langle \{\bullet_{[t]} \mid t \in \mathscr{T}_\Sigma(X)_\mathfrak{s}\} \rangle_{\mathfrak{s} \in \mathscr{B}(S)}$ such that for all $\mathfrak{s} \in \mathscr{B}(S)$ and $x \in X_\mathfrak{s}$ we have $\alpha(x) = \bullet_{[\sigma(x)]}$. For every substitution $\sigma : X \to \mathscr{T}_\Sigma(X)$ and $t \in \mathscr{T}_\Sigma(X)_\mathfrak{s}$, we write $\sigma(t)$ or $t\sigma$ for $\sigma_\mathfrak{s}(t)$. A substitution $\sigma : X \to \mathscr{T}_\Sigma(X)$ is *closed* if $\sigma(x) \in \mathscr{C}\mathscr{T}_\Sigma(X)$ for all $x \in X$. As in case of ordinary substitutions, we write $[t_1, \ldots, t_n/x_1, \ldots, x_n]$ for the substitution $\sigma$ such that $\sigma(x_1) = t_1$, ..., $\sigma(x_n) = t_n$ and $\sigma(x) = x$ if $x \notin \{x_1, \ldots, x_n\}$.

Substitution extends ordinary substitution from ordinary variables to all variables. Notice that substitution is defined up to $\dot{\approx}$-equivalence explicitly. Notice further that

$$\sigma(x) = x_1, \ldots, x_n \,.\, t \Rightarrow \sigma(x(t_1, \ldots, t_n)) \mathrel{\dot{\approx}} t[\sigma(t_1), \ldots, \sigma(t_n)/x_1, \ldots, x_n].$$

**Example 2.17.** Consider again the binding signature $\Sigma$ used in Example 2.10 and suppose that $X$ is a variable domain for $\Sigma$. Let $x, z \in X$ be such that $x : \mathsf{P}$ and $z : \mathsf{P} \,.\, \mathsf{P}$. Then $\mu x \,.\, z(x)$ and $z(\mu x \,.\, z(x))$ are binding terms over $\Sigma$ and $X$ with free variable $z$. Substitution of $x \,.\, ax$ for $z$ in these terms yields the following results: $\mu x \,.\, z(x)[x \,.\, ax/z] \mathrel{\dot{\approx}} \mu x \,.\, ax$ and $z(\mu x \,.\, z(x))[x \,.\, ax/z] \mathrel{\dot{\approx}} a\mu x \,.\, ax$.

### *2.3. Transition system specifications*

In this section, we generalize the notion of TSS from conventional terms to binding terms. The meaning of TSSs is discussed in Section 3.

The TSSs of Ref. [24], which originate from Refs. [35], define binary transition relations by means of transition rules with positive premises. An extension to transi-

tion rules with positive and negative premises was presented in Refs. [10,22] and a further extension to the specification of unary and binary relations was presented in Ref. [38]. In all three cases, variable binding operators and many-sortedness are not supported. Such an extension was first proposed in Ref. [17]. An important difference between that extension and the one presented here is that in the latter distinction between formal and actual variables, formal and actual terms, formal and actual substitutions, etc., is not made. This leads to TSSs that are more closely related to the original ones than the TSSs of Ref. [17].

We define TSSs in terms of transition rules and transition rules in terms of transition formulas. We define transition formulas over a binding signature and a domain of transition predicates. Therefore, we first define the notion of domain of transition predicates.

We assume a $\mathscr{B}(\mathscr{S})^*$-indexed family of mutually disjoint sets of *predicates* $\mathscr{P} = \langle \mathscr{P}_\tau \rangle_{\tau \in \mathscr{B}(\mathscr{S})^*}$. It is assumed that the sets $\bigcup_{\mathfrak{s} \in \mathscr{B}(\mathscr{S})} \mathscr{V}_\mathfrak{s}$, $\bigcup_{\tau \in \mathscr{B}(\mathscr{S})^* \times \mathscr{S}} \mathscr{O}_\tau$ and $\bigcup_{\tau \in \mathscr{B}(\mathscr{S})^*} \mathscr{P}_\tau$ are mutually disjoint. We write $p : \mathfrak{s}_1 \times \cdots \times \mathfrak{s}_n$ to indicate that $p \in \mathscr{P}_{\langle \mathfrak{s}_1, \ldots, \mathfrak{s}_n \rangle}$.

A predicate $p : \mathfrak{s}_1 \times \cdots \times \mathfrak{s}_n$ has $n$ arguments. If $\mathfrak{s}_i = s_{i1}, \ldots, s_{in_i} \,.\, s_i$ $(1 \leqslant i \leqslant n)$, then it binds $n_i$ variables, of base sorts $s_{i1}, \ldots, s_{in_i}$, in the $i$th argument. Otherwise, i.e., if $\mathfrak{s}_i \in S$, it does not bind any variable in the $i$th argument.

**Definition 2.18.** Let $\Sigma = (S, O)$ be a binding signature. Then a *domain of transition predicates* on terms over $\Sigma$ is a set $\Pi \subseteq \mathscr{P}$ such that for all $p \in \Pi$, if $p : \mathfrak{s}_1 \times \cdots \times \mathfrak{s}_n$, then $\mathfrak{s}_1, \ldots, \mathfrak{s}_n \in \mathscr{B}(S)$, $\mathfrak{s}_1 \in S$ and $1 \leqslant n \leqslant 2$. A transition predicate $p \in \Pi$ is called *ordinary* if $p : \mathfrak{s}_1 \times \cdots \times \mathfrak{s}_n$ for some $\mathfrak{s}_1, \ldots, \mathfrak{s}_n \in S$.

Just as in Ref. [38], we consider both unary and binary predicates as transition predicates. The restriction that a transition predicate is a unary or binary predicate is formulated here to anticipate its relaxation in Section 5. We do not consider predicates that bind variables in their first argument as transition predicates. The main reason for this exclusion is that we cannot conceive of an obvious generalization of the notion of bisimulation in case variables are bound in the first argument.

Next, in Definition 2.19, we define the notions of positive and negative transition formula. We also introduce the notion of denial of a transition formula and make the notion of closed transition formula precise. Like in Refs. [10,22,38], we consider both positive and negative transition formulas. The formation rule for negative formulas does not allow a negative formula of the form $\neg p(t_1)$ for binary predicates $p$, i.e., predicates $p$ with $p : \mathfrak{s}_1 \times \mathfrak{s}_2$ for some sorts $\mathfrak{s}_1$ and $\mathfrak{s}_2$. In Refs. [10,22,38], such expressions are considered to be negative formulas. We consider them to be abbreviations of sets of negative formulas (see also Definition 2.23).

**Definition 2.19.** Let $\Pi$ be a domain of transition predicates on terms over binding signature $\Sigma = (S, O)$. Then $\mathscr{F}^+_{\Sigma,\Pi}$, the set of *positive transition formulas* over signature $\Sigma$ and transition predicates $\Pi$, is inductively defined by the following formation rule $(1 \leqslant n \leqslant 2)$:

$$p \in \Pi,\ \ p : \mathfrak{s}_1 \times \cdots \times \mathfrak{s}_n, t_1 \in \mathscr{T}_{\Sigma_{\mathfrak{s}_1}}, \ldots, t_n \in \mathscr{T}_{\Sigma_{\mathfrak{s}_n}} \ \ \Rightarrow \ \ p(t_1, \ldots, t_n) \in \mathscr{F}^+_{\Sigma,\Pi};$$

and $\mathscr{F}^-_{\Sigma,\Pi}$, the set of *negative transition formulas* over signature $\Sigma$ and transition predicates $\Pi$, is inductively defined by the following formation rule $(1 \leqslant n \leqslant 2)$:

$$p \in \Pi,\ \ p : \mathfrak{s}_1 \times \cdots \times \mathfrak{s}_n, t_1 \in \mathscr{T}_{\Sigma_{\mathfrak{s}_1}}, \ldots, t_n \in \mathscr{T}_{\Sigma_{\mathfrak{s}_n}} \ \Rightarrow\ \neg p(t_1, \ldots, t_n) \in \mathscr{F}^-_{\Sigma,\Pi}.$$

We use in general postfix notation for unary predicates and infix notation for binary predicates. We write $\mathscr{F}_{\Sigma,\Pi}$ for $\mathscr{F}^+_{\Sigma,\Pi} \cup \mathscr{F}^-_{\Sigma,\Pi}$. For $\phi \in \mathscr{F}_{\Sigma,\Pi}$, $\overline{\phi}$, the *denial* of $\phi$, is defined as follows:

1. $\overline{p(t_1, \ldots, t_m)} = \neg p(t_1, \ldots, t_m)$;
2. $\overline{\neg p(t_1, \ldots, t_m)} = p(t_1, \ldots, t_m)$.

A positive or negative transition formula $\phi$ is *closed* if all terms occurring in it are closed. We write $\mathscr{CF}^+_{\Sigma,\Pi}$ for $\{\phi \in \mathscr{F}^+_{\Sigma,\Pi} \mid \phi \text{ is closed}\}$ and $\mathscr{CF}^-_{\Sigma,\Pi}$ for transition formulas $\{\phi \in \mathscr{F}^-_{\Sigma,\Pi} \mid \phi \text{ is closed}\}$. Furthermore, we write $\mathscr{CF}_{\Sigma,\Pi}$ for $\mathscr{CF}^+_{\Sigma,\Pi} \cup \mathscr{CF}^-_{\Sigma,\Pi}$.

**Example 2.20.** In previous examples, we used $\mu x\,.\,ax$ as an example of a closed binding term. Suppose that we have a transition predicate $\xrightarrow{a} : \mathsf{P} \times \mathsf{P}$. The intended meaning of a transition formula of the form $t \xrightarrow{a} t'$ can be explained as follows: process $t$ is capable of first performing action $a$ and then proceeding as process $t'$. Hence, the transition formula $\mu x\,.\,ax \xrightarrow{a} \mu x\,.\,ax$ expresses that $\mu x\,.\,ax$ is capable of performing action $a$ forever.

In the following definition, the notion of transition rule is defined. Like in Refs. [10,22,38], negative formulas are not allowed as conclusions of transition rules. The notions of substitution instance and closed substitution instance of a transition rule are also introduced.

**Definition 2.21.** Let $\Pi$ be a domain of transition predicates on terms over binding signature $\Sigma = (S, O)$. Then $\mathscr{R}_{\Sigma,\Pi}$, the set of *transition rules* over signature $\Sigma$ and predicates $\Pi$, is inductively defined by the following formation rule:

$$\Phi \subseteq \mathscr{F}_{\Sigma,\Pi},\quad \psi \in \mathscr{F}^+_{\Sigma,\Pi} \ \Rightarrow\ \frac{\Phi}{\psi} \in \mathscr{R}_{\Sigma,\Pi}.$$

Let $r = \frac{\Phi}{\psi}$ be a transition rule. Then the transition formulas in $\Phi$ are the *premises* of $r$ and the transition formula $\psi$ is the *conclusion* of $r$. A transition rule $r$ is *closed* if all formulas occurring in it are closed. Substitution extends from terms to formulas and rules as expected. For every substitution $\sigma : \mathscr{V} \to \mathscr{T}_\Sigma$ and transition rule $r$, the transition rule $\sigma(r)$ is a *substitution instance* of $r$. If $\sigma$ is a closed substitution, the transition rule $\sigma(r)$ is a *closed* substitution instance of $r$. We write *instances*$(r)$ for the set of all substitution instances of $r$, and *cinstances*$(r)$ for the set of all closed substitution instances of $r$.

**Example 2.22.** The transition rule for the recursion operator of CCS is as follows:

$$\frac{z(\mu x\,.\,z(x)) \xrightarrow{\alpha} x'}{\mu x\,.\,z(x) \xrightarrow{\alpha} x'}.$$

Finally, the notion of TSS is defined. The main difference with the original notion of TSS is that binding terms are used instead of conventional terms. This means not only that operators may bind variables in their arguments, but also that variables may have arguments. Having variables that may have arguments obviates the need

to distinguish two kinds of variables, terms, substitutions, etc., in TSSs, like in Ref. [17].

**Definition 2.23.** A TSS is a triple $P = (\Sigma, \Pi, R)$, where $\Sigma$ is a binding signature, $\Pi$ is a domain of transition predicates on terms over $\Sigma$ and $R \subseteq \mathscr{R}_{\Sigma,\Pi}$. A TSS is *positive* if all premises of its transition rules are positive transition formulas. We write $\mathit{instances}(R)$ for the set $\{\mathit{instances}(r) \mid r \in R\}$, and $\mathit{cinstances}(R)$ for the set $\{\mathit{cinstances}(r) \mid r \in R\}$. For each $p \in \Pi$, $p : \mathfrak{s}_1 \times \mathfrak{s}_2$, and $t_1 \in \mathscr{T}_{\Sigma_{\mathfrak{s}_1}}$, we write $\neg p(t_1)$ for the set of formulas $\{\neg p(t_1, t_2) \mid t_2 \in \mathscr{CT}_{\Sigma_{\mathfrak{s}_2}}\}$.

Recall that, unlike in Refs. [10,22,38], an expression of the form $\neg p(t_1)$ is not considered to be a negative formula if $p$ is a binary predicate. Instead, it is considered to be an abbreviation of the set of formulas that contains all formulas $\neg p(t_1, t_2)$ where $t_2$ is a closed term of the appropriate sort. This leads to some simpler definitions in the remaining sections.

**Example 2.24.** We consider a fragment of CCS without restriction and relabeling, but with recursion. CCS assumes a set $\mathsf{N}$ of names. The set $\mathsf{A}$ of actions is defined by $\mathsf{A} = \mathsf{N} \cup \overline{\mathsf{N}} \cup \{\tau\}$, where $\overline{\mathsf{N}} = \{\overline{a} \mid a \in \mathsf{N}\}$. Elements $\overline{a} \in \overline{\mathsf{N}}$ are called co-names and $\tau$ is called the silent step. The signature of the TSS for this fragment of CCS consists of the sort $\mathsf{P}$ of processes, the inaction constant $0 :\to \mathsf{P}$, an action prefix operator $\alpha : \mathsf{P} \to \mathsf{P}$ for each action $\alpha \in \mathsf{A}$, the choice operator $+ : \mathsf{P} \times \mathsf{P} \to \mathsf{P}$, the composition operator $| : \mathsf{P} \times \mathsf{P} \to \mathsf{P}$, and the recursion (variable binding) operator $\mu : \mathsf{P} \,.\, \mathsf{P} \to \mathsf{P}$. The transition predicate domain consists of a binary transition predicate $\xrightarrow{\alpha} : \mathsf{P} \times \mathsf{P}$ for each $\alpha \in \mathsf{A}$. The transition rules are the transition rules given below and the transition rule given in Example 2.22 ($\alpha \in \mathsf{A}$, $a \in \mathsf{N}$).

$$\frac{}{\alpha x \xrightarrow{\alpha} x} \quad \frac{x \xrightarrow{\alpha} x'}{x + y \xrightarrow{\alpha} x'} \quad \frac{y \xrightarrow{\alpha} y'}{x + y \xrightarrow{\alpha} y'}$$

$$\frac{x \xrightarrow{\alpha} x'}{x \mid y \xrightarrow{\alpha} x' \mid y} \quad \frac{y \xrightarrow{\alpha} y'}{x \mid y \xrightarrow{\alpha} x \mid y'} \quad \frac{x \xrightarrow{a} x', y \xrightarrow{\overline{a}} y'}{x \mid y \xrightarrow{\tau} x' \mid y'} \quad \frac{x \xrightarrow{\overline{a}} x', y \xrightarrow{a} y'}{x \mid y \xrightarrow{\tau} x' \mid y'}.$$

## 3. The meaning of TSSs

In this section, we first introduce the basic notions relevant to the issue of associating models with TSSs. These basic notions are sufficient in case of positive TSSs. Next, we discuss the principle underlying the association of a model with a positive TSS. Finally, using this principle as a guideline, we introduce the additional notions relevant to the issue of associating models with TSSs that are not positive.

The meaning of TSSs with negative premises has been extensively studied for TSSs that define transition relations on conventional terms, see, e.g., Refs. [1,10,21,22,38]. Most definitions and results generalize naturally to the case of TSSs that define transition relations on $\doteq$-equivalence classes of binding terms, which will henceforth often loosely be referred to as transition relations on binding terms. However, the smooth generalization is perhaps not apparent because the presentation of the material is new here. All work on the meaning of TSSs with negative premises uses

many results of work on logic programming with negation. An excellent survey of relevant work in that area is Ref. [3].

### 3.1. Supported models

We define the notions of model of a TSS and interpretation supported by a TSS. This requires to introduce first a kind of structures in which transition rules can be interpreted. We introduce two equivalent kinds: transition systems and Herbrand interpretations. However, in the sequel, we will focus on Herbrand interpretations.

The most obvious choice of structures for the interpretation of transition rules is probably the choice of transition systems. They are defined here with respect to a TSS, but they can alternatively be defined with respect to a binding signature and a domain of transition predicates.

**Definition 3.1.** Let $P = (\Sigma, \Pi, R)$ be a TSS. A *transition system* for $P$ is a family of relations $\mathscr{J} = \langle \mathscr{J}_p \rangle_{p \in \Pi}$, where

$$\text{for each } p \in \Pi, p : \mathfrak{s}_1 \times \cdots \times \mathfrak{s}_n, \mathscr{J}_p \text{ is a relation}$$
$$\mathscr{J}_p \subseteq [\mathscr{CT}_{\Sigma_{\mathfrak{s}_1}}] \times \cdots \times [\mathscr{CT}_{\Sigma_{\mathfrak{s}_n}}], \text{ called the } \textit{denotation} \text{ of } p.$$

So predicates are interpreted as relations on sets of $\dot{\approx}$-equivalence classes of closed binding terms. This makes precise that we identify binding terms that can be obtained from each other by change of bound variables. This identification of binding terms induces the following identification of transition formulas.

**Definition 3.2.** Let $\Pi$ be a domain of transition predicates on terms over binding signature $\Sigma = (S, O)$. Then $\dot{\approx}$ is the least equivalence relation $\dot{\approx} \subseteq \mathscr{F}_{\Sigma,\Pi} \times \mathscr{F}_{\Sigma,\Pi}$ such that

$$t_1 \dot{\approx} t'_1, \ldots, t_n \dot{\approx} t'_n, p : \mathrm{s}(t_1) \times \cdots \times \mathrm{s}(t_n)$$
$$\Rightarrow p(t_1, \ldots, t_n) \dot{\approx} p(t'_1, \ldots, t'_n), \neg p(t_1, \ldots, t_n) \dot{\approx} \neg p(t'_1, \ldots, t'_n).$$

Furthermore, let $\phi \in \mathscr{F}_{\Sigma,\Pi}$ and $\Phi \subseteq \mathscr{F}_{\Sigma,\Pi}$. Then we write $[\phi]$ for $\{\psi \in \mathscr{F}_{\Sigma,\Pi} \mid \phi \dot{\approx} \psi\}$, and $[\Phi]$ for $\{[\phi] \mid \phi \in \Phi\}$.

Another choice of structures for the interpretation of transition rules, customary in logic programming, is the choice of Herbrand interpretations.

**Definition 3.3.** Let $P = (\Sigma, \Pi, R)$ be a TSS. A *Herbrand interpretation* for $P$ is a set $\mathscr{M} \subseteq \mathscr{CF}^+_{\Sigma,\Pi}$ such that $[\mathscr{M}] \subseteq \mathscr{M}$.

An Herbrand interpretation is simply a set of transition formulas. The condition $[\mathscr{M}] \subseteq \mathscr{M}$ implies that either all transition formulas from the same $\dot{\approx}$-equivalence class are in a Herbrand interpretation or none is. Thus, it is guaranteed that there exists a bijection between the class of Herbrand interpretations for $P$ and the class of transition systems for $P$: a Herbrand interpretation $\mathscr{M}$ corresponds to the transition system $\mathscr{J} = \langle \mathscr{J}_p \rangle_{p \in \Pi}$ such that

$$\text{for each } p \in \Pi, \quad \mathscr{J}_p = \{([t_1], \ldots, [t_n]) \mid p(t_1, \ldots, t_n) \in \mathscr{M}\}.$$

Hence, a transition relation on binding terms over $\Sigma$ can be regarded as a set of closed positive transition formulas over $\Sigma$. Therefore, closed positive transition formulas will sometimes loosely be referred to as *transitions*. Likewise, closed negative transition formulas will sometimes be referred to as *negative transitions*. Because of the existence of a bijection between the class of Herbrand interpretations for a TSS and the class of transition systems for it, we can safely focus on Herbrand interpretations. The latter structures make it easier to explain what model is associated with a TSS that is not positive.

Before we can define what Herbrand interpretations for a TSS are models of that TSS and what Herbrand interpretations for a TSS are supported by that TSS, we have to make precise what it means for a transition formula to hold in a Herbrand interpretation. The following definition states that a positive transition formula holds in a Herbrand interpretation if it is contained in that Herbrand interpretation and a negative transition formula holds in a Herbrand interpretation if its denial is not contained in that Herbrand interpretation.

**Definition 3.4.** Let $\mathcal{M}$ be a Herbrand interpretation for a TSS $P = (\Sigma, \Pi, R)$. Furthermore, let $\phi \in \mathcal{CF}_{\Sigma,\Pi}$. Then $\phi$ *holds* in $\mathcal{M}$, written $\mathcal{M} \models \phi$, if
1. either $\phi \in \mathcal{CF}^+_{\Sigma,\Pi}$ and $\phi \in \mathcal{M}$;
2. or $\phi \in \mathcal{CF}^-_{\Sigma,\Pi}$ and $\overline{\phi} \notin \mathcal{M}$.
For $\Phi \subseteq \mathcal{CF}_{\Sigma,\Pi}$, we write $\mathcal{M} \models \Phi$ to indicate that $\mathcal{M} \models \phi$ for all $\phi \in \Phi$.

Notice that $\mathcal{M} \models \phi$ iff $\mathcal{M} \models [\phi]$. Next, in Definitions 3.5 and 3.6, we define the notions of model of a TSS and interpretation supported by a TSS. Roughly speaking, these definitions express that a Herbrand interpretation for a TSS is a model of that TSS if it obeys the transition rules of the TSS and that a Herbrand interpretation for a TSS is supported by that TSS if all transitions contained in it are justified by the transition rules of the TSS.

**Definition 3.5.** Let $\mathcal{M}$ be a Herbrand interpretation for a TSS $P = (\Sigma, \Pi, R)$. Then $\mathcal{M}$ is a *Herbrand model* of $P$, written $\mathcal{M} \models P$, if for all $\psi \in \mathcal{CF}^+_{\Sigma,\Pi}$:

$$\mathcal{M} \models \psi \;\Leftarrow\; \exists \frac{\Phi}{\psi} \in \mathit{cinstances}(R) \bullet \mathcal{M} \models \Phi.$$

**Definition 3.6.** Let $\mathcal{M}$ be a Herbrand interpretation for a TSS $P = (\Sigma, \Pi, R)$. Then $\mathcal{M}$ is *supported* by $P$ if for all $\psi \in \mathcal{CF}^+_{\Sigma,\Pi}$:

$$\mathcal{M} \models \psi \;\Rightarrow\; \exists \frac{\Phi}{\psi} \in \mathit{cinstances}(R) \bullet \mathcal{M} \models \Phi.$$

Finally, we define what Herbrand interpretations for a TSS agree with that TSS. This notion of agreeing with a TSS, which is used in, e.g., Refs. [21,22,38], is also known as "being a supported model of a TSS".

**Definition 3.7.** Let $\mathcal{M}$ be a Herbrand interpretation for a TSS $P = (\Sigma, \Pi, R)$. Then $\mathcal{M}$ *agrees with* $P$ if $\mathcal{M}$ is a Herbrand model of $P$ and $\mathcal{M}$ is supported by $P$. If $\mathcal{M}$ is a Herbrand interpretation that agrees with $P$, we say that $\mathcal{M}$ is a *supported* model of $P$.

**Example 3.8.** Consider the simple TSS with a one-sorted signature consisting of a constant $c$ only, two transition predicates $\stackrel{a}{\rightarrow}$ and $\stackrel{b}{\rightarrow}$, and the following transition rules:

$$\frac{c \stackrel{a}{\not\rightarrow}}{c \stackrel{b}{\rightarrow} c} \qquad \frac{c \stackrel{a}{\rightarrow} c}{c \stackrel{a}{\rightarrow} c}$$

As usual, we use the notation $c \stackrel{a}{\not\rightarrow}$ instead of $\neg(c \stackrel{a}{\rightarrow})$. Both $\{c \stackrel{a}{\rightarrow} c\}$ and $\{c \stackrel{b}{\rightarrow} c\}$ agree with this TSS.

Every positive TSS has a least supported model with respect to set inclusion. The least supported model of a positive TSS has two interesting alternative characterizations, which will be given in Section 3.2.

### *3.2. Proofs and positive TSSs*

We define a general notion of proof from a TSS by allowing to prove transition rules. The proof of a transition rule $\frac{\Phi}{\psi}$ corresponds to the proof of the transition formula $\psi$ under the assumptions $\Phi$. In Section 3.3, it happens that allowing to prove transition rules is quite useful.

**Definition 3.9.** Let $P = (\Sigma, \Pi, R)$ be a TSS. Then a *proof* of a transition rule $\frac{\Phi}{\psi}$ from $P$ is a well-founded, upwardly branching tree of which the nodes are labelled by formulas in $\mathscr{F}_{\Sigma,\Pi}$, such that

1. the root is labelled by $\psi$;
2. if a node is labelled by $\phi$ and $\Phi'$ is the set of labels of the nodes directly above this node, then there exists a $\psi'$ such that $\phi \mathrel{\dot{\approx}} \psi'$ and
   (a) either $\psi' \in \Phi$ and $\Phi' = \emptyset$,
   (b) or $\frac{\Phi'}{\psi'} \in \mathit{instances}(R)$.

A transition rule $r$ is *provable* from $P$, written $P \vdash r$, if there exists a proof of $r$ from $P$. A positive transition formula $\phi$ is *provable* from $P$, written $P \vdash \phi$, if there exists a proof of $\frac{\emptyset}{\phi}$ from $P$.

**Example 3.10.** In Example 2.20, we used $\mu x \,.\, ax \stackrel{a}{\rightarrow} \mu x \,.\, ax$ as an example of a transition formula. The proof of $\mu x \,.\, ax \stackrel{a}{\rightarrow} \mu x \,.\, ax$ is as follows:

$$\begin{array}{l} a\mu x \,.\, ax \stackrel{a}{\rightarrow} \mu x \,.\, ax \\ \mu x \,.\, ax \stackrel{a}{\rightarrow} \mu x \,.\, ax \end{array}$$

The non-root node is obtained from the first rule given in Example 2.24 with substitution of $\mu x \,.\, ax$ for $x$. The root node is obtained from the non-root node and the rule for the recursion operator given in Example 2.22 with substitution of $x \,.\, ax$ for $z$ and $\mu x \,.\, ax$ for $x'$ (see also Example 2.17).

It is easy to see that, if $\phi \in \Phi \iff \exists \phi' \in \Phi' \bullet \phi \mathrel{\dot{\approx}} \phi'$, $\phi' \in \Phi' \iff \exists \phi \in \Phi \bullet \phi' \mathrel{\dot{\approx}} \phi$ and $\psi \mathrel{\dot{\approx}} \psi'$, then $P \vdash \frac{\Phi}{\psi} \iff P \vdash \frac{\Phi'}{\psi'}$. We have the following soundness result:

$$\text{If } P \vdash \frac{\Phi}{\psi}, \text{ then for all Herbrand models } \mathscr{M} \text{ of } P, \mathscr{M} \models \Phi \Rightarrow \mathscr{M} \models \psi.$$

The intended model of a positive TSS reflects the idea that the following principle implicitly applies to a TSS: "the only transition formulas that hold in the intended model are those derivable from the transition rules."

**Definition 3.11.** The *intended* Herbrand model of a positive TSS $P = (\Sigma, \Pi, R)$, written $\mathscr{M}_P$, is the Herbrand model $\{\phi \in \mathscr{CF}^+_{\Sigma,\Pi} \mid P \vdash \phi\}$.

Clearly, every positive TSS has a unique intended Herbrand model. Moreover, the intended Herbrand model of a positive TSS is the least supported model of that TSS. The intended model of a positive TSS can also be characterized by means of the immediate consequence operator originating from Ref. [13]. The immediate consequence operator for a TSS $P$ applied to a Herbrand interpretation $\mathscr{M}$ yields the smallest Herbrand interpretation containing all closed positive transition formulas that are immediate consequences of the closed transition formulas that hold in $\mathscr{M}$ and the transition rules of $P$.

**Definition 3.12.** Let $P = (\Sigma, \Pi, R)$ be a TSS. Then the *immediate consequence operator* for $P$ is the unary function $\mathbf{T}_P$ on Herbrand interpretations for $P$ such that for all Herbrand interpretations $\mathscr{M}$ for $P$:

$$\mathbf{T}_P(\mathscr{M}) = \bigcup \left\{ [\psi] \mid \psi \in \mathscr{CF}^+_{\Sigma,\Pi} \text{ and } \exists \frac{\Phi}{\psi} \in \mathit{cinstances}(R) \bullet \mathscr{M} \models \Phi \right\}.$$

For all Herbrand interpretations $\mathscr{M}$ for a positive TSS $P = (\Sigma, \Pi, R)$:
1. $\mathbf{T}_P(\mathscr{M}) \subseteq \mathscr{M} \iff \mathscr{M}$ is a Herbrand model of $P$,
2. $\mathbf{T}_P(\mathscr{M}_P) = \mathscr{M}_P$;
i.e., $\mathscr{M}$ is a Herbrand model of $P$ iff $\mathscr{M}$ is closed under $\mathbf{T}_P$ and the intended Herbrand model $\mathscr{M}_P$ is the least Herbrand interpretation closed under $\mathbf{T}_P$ – as well as the least supported model of $P$. In case $P$ is not positive, the existence of a least Herbrand interpretation closed under $\mathbf{T}_P$ – and also of a least supported model of $P$ – is not guaranteed.

**Example 3.13.** Consider the simple positive TSS $P$ with a one-sorted signature consisting of a constant $c$ and a unary operator $f$ only, a transition predicate $\xrightarrow{a}$, and the following transition rules:

$$\frac{}{c \xrightarrow{a} c} \qquad \frac{c \xrightarrow{a} y}{c \xrightarrow{a} f(y)}$$

According to Definition 3.11, $\mathscr{M}_P = \{c \xrightarrow{a} f^n(c) \mid n \geqslant 0\}$. Obviously, we have $\mathbf{T}_P(\mathscr{M}_P) = \mathscr{M}_P$.

### 3.3. Stable models

In case a TSS is not positive, it is possible that proofs exist for transition rules $\frac{N}{\psi}$, where $N$ is a non-empty set of closed negative transition formulas and $\psi$ is a closed positive transition formula. Such proofs can never be extended to proofs of the conclusions concerned because no rule of a TSS has a negative transition formula as its conclusion. This means that for the intended model of a TSS that is not positive, it is

reasonable to adapt the principle that implicitly applies to a TSS as follows: "the only transition formulas that hold in the intended model are those derivable from the transition rules under assumption of negative transition formulas that do not lead to inconsistencies." In order to formalize this principle, it is useful to introduce an operator that replaces in a TSS the original transition rules by the provable closed transition rules without positive premises.

**Definition 3.14.** Let $P = (\Sigma, \Pi, R)$ be a TSS. Then the TSS $P^*$ is defined as $(\Sigma, \Pi, R^*)$, where

$$R^* = \left\{ \frac{N}{\psi} \mid N \subseteq \mathscr{CF}^-_{\Sigma,\Pi},\ \psi \in \mathscr{CF}^+_{\Sigma,\Pi} \text{ and } P \vdash \frac{N}{\psi} \right\}.$$

Adopting the above-mentioned principle, we conclude that the intended model of a TSS $P = (\Sigma, \Pi, R)$ must be a model $\mathscr{M}$ such that $\mathbf{T}_{P^*}(\mathscr{M}) \subseteq \mathscr{M}$. Such a model is called a *stable* model of $P$. Stable models were first introduced in Ref. [20] to give a semantics of logic programming with negation.

However, even in case the adapted principle is applied, it is possible that there remain closed positive transition formulas $\phi$ for which it is not possible to decide whether $\phi$ holds in the intended model or not. In such cases, the TSS concerned is called *incomplete*. Clearly, an incomplete TSS does not have a (stable) model that can be designated as its intended model. In other words, an incomplete TSS does not have a least stable model, i.e., a model $\mathscr{M}$ such that $\mathbf{T}_{P^*}(\mathscr{M}) = \mathscr{M}$. Besides, an unsound transition rule, i.e., a rule with a premise that contradicts the conclusion, is simply ignored in case the adapted principle is applied. For these reasons, we use the auxiliary notion of partial Herbrand interpretation to define the intended model of a TSS that is not positive.

**Example 3.15.** Consider the simple TSS with a one-sorted signature consisting of a constant $c$ only, two transition predicates $\xrightarrow{a}$ and $\xrightarrow{b}$, and the following transition rules:

$$\frac{c \xrightarrow{a}}{c \xrightarrow{b} c} \qquad \frac{c \xrightarrow{b}}{c \xrightarrow{a} c} \qquad \frac{c \xrightarrow{a}}{c \xrightarrow{a} c}$$

The Herbrand interpretation $\{c \xrightarrow{a} c\}$ is a stable model of this TSS, although the premise and conclusion of the last transition rule contradict each other.

**Definition 3.16.** Let $P = (\Sigma, \Pi, R)$ be a TSS. A *partial* Herbrand interpretation for $P$ is a pair $\mathscr{M} = (\mathscr{M}^+, \mathscr{M}^-)$, with sets $\mathscr{M}^+, \mathscr{M}^- \subseteq \mathscr{CF}^+_{\Sigma,\Pi}$ such that $[\mathscr{M}^+] \subseteq \mathscr{M}^+$ and $[\mathscr{M}^-] \subseteq \mathscr{M}^-$. A partial Herbrand interpretation $\mathscr{M}$ for $P$ is *consistent* if $\mathscr{M}^+ \cap \mathscr{M}^- = \emptyset$. A partial Herbrand interpretation $\mathscr{M}$ for $P$ is *total* if $\mathscr{M}^+ \cup \mathscr{M}^- = \mathscr{CF}^+_{\Sigma,\Pi}$. Let $\mathscr{M} = (\mathscr{M}^+, \mathscr{M}^-)$ and $\mathscr{N} = (\mathscr{N}^+, \mathscr{N}^-)$ be partial Herbrand interpretations for $P$. Then $\mathscr{M} \subseteq \mathscr{N}$ iff $\mathscr{M}^+ \subseteq \mathscr{N}^+$ and $\mathscr{M}^- \subseteq \mathscr{N}^-$.

The intuition is that the positive component $\mathscr{M}^+$ contains the transitions that certainly hold and the negative component $\mathscr{M}^-$ contains the transitions that certainly do not hold. Thus, a unique partial Herbrand model can be associated with all TSSs, even the incomplete ones. The conditions $[\mathscr{M}^+] \subseteq \mathscr{M}^+$ and $[\mathscr{M}^-] \subseteq \mathscr{M}^-$

imply that either all transition formulas from the same $\dot{\approx}$-equivalence class are in one of the two components of a partial Herbrand interpretation or none is. Obviously, every Herbrand interpretation $\mathcal{M}$ for a TSS $P = (\Sigma, \Pi, R)$ can be identified with the total, consistent partial Herbrand interpretation $(\mathcal{M}, \mathcal{CF}^+_{\Sigma,\Pi} \setminus \mathcal{M})$. The ordering $\subseteq$ on partial Herbrand interpretations is called the information ordering on partial Herbrand interpretations: if $\mathcal{M} \subseteq \mathcal{N}$, then the transitions about which $\mathcal{N}$ contains status information include the transitions about which $\mathcal{M}$ contains status information. Partial Herbrand interpretations were first introduced in Ref. [15] in the context of logic programming. They are also known as three-valued Herbrand interpretations.

We will define an immediate consequence operator on partial Herbrand interpretations as well. Before we can do so, we have to make precise what it means for a transition formula to hold in a partial Herbrand interpretation. The following definition states that a positive transition formula holds in a partial Herbrand interpretation if it is contained in the positive component of that partial Herbrand interpretation and a negative transition formula holds in a partial Herbrand interpretation if its denial is contained in the negative component of that partial Herbrand interpretation.

**Definition 3.17.** Let $\mathcal{M} = (\mathcal{M}^+, \mathcal{M}^-)$ be a partial Herbrand interpretation for a TSS $P = (\Sigma, \Pi, R)$. In addition, let $\phi \in \mathcal{CF}_{\Sigma,\Pi}$. Then $\phi$ *holds* in $\mathcal{M}$, written $\mathcal{M} \models_3 \phi$, if
1. either $\phi \in \mathcal{CF}^+_{\Sigma,\Pi}$ and $\phi \in \mathcal{M}^+$;
2. or $\phi \in \mathcal{CF}^-_{\Sigma,\Pi}$ and $\overline{\phi} \in \mathcal{M}^-$.
For $\Phi \subseteq \mathcal{CF}_{\Sigma,\Pi}$, we write $\mathcal{M} \models_3 \Phi$ to indicate that $\mathcal{M} \models_3 \phi$ for all $\phi \in \Phi$. Furthermore, we write $\mathcal{M} \overline{\models}_3 \Phi$ to indicate that $\mathcal{M} \models_3 \overline{\phi}$ for some $\phi \in \Phi$.

In Definition 3.18, we define an immediate consequence operator on partial Herbrand interpretations. This operator takes into account that a TSS may be incomplete: it yields both the closed positive transition formulas of which it can be decided that they are immediate consequences and the ones of which it can be decided that they are not immediate consequences.

**Definition 3.18.** Let $P = (\Sigma, \Pi, R)$ be a TSS. Then the *immediate consequence operator* for $P$ for partial Herbrand interpretations is the unary function $\mathbf{T}^3_P$ on partial Herbrand interpretations for $P$ such that for all partial Herbrand interpretations $\mathcal{M} = (\mathcal{M}^+, \mathcal{M}^-)$ for $P$:

$$\mathbf{T}^3_P(\mathcal{M}) = (\mathcal{N}^+, \mathcal{N}^-) \text{ where}$$
$$\mathcal{N}^+ = \bigcup \left\{ [\psi] \mid \psi \in \mathcal{CF}^+_{\Sigma,\Pi} \text{ and } \exists \frac{\Phi}{\psi} \in \mathit{cinstances}(R) \bullet \mathcal{M} \models_3 \Phi \right\};$$
$$\mathcal{N}^- = \bigcup \left\{ [\psi] \mid \psi \in \mathcal{CF}^+_{\Sigma,\Pi} \text{ and } \forall \frac{\Phi}{\psi} \in \mathit{cinstances}(R) \bullet \mathcal{M} \overline{\models}_3 \Phi \right\}.$$

**Example 3.19.** Consider the simple TSS $P$ with a one-sorted signature consisting of a constant $c$ and a unary operator $f$ only, a transition predicate $\xrightarrow{a}$, and the following transition rules:

$$\frac{}{c \xrightarrow{a} c} \qquad \frac{x \overset{a}{\nrightarrow} c}{f(x) \xrightarrow{a} c}$$

The partial Herbrand interpretation $(\{f^{2n}(c) \xrightarrow{a} c \mid n \geqslant 0\}, \{f^{2n+1}(c) \xrightarrow{a} c \mid n \geqslant 0\})$ is the least partial Herbrand interpretation, with respect to the information ordering, closed under $\mathbf{T}^3_P$. It is also consistent and total.

Using Definitions 3.14 and 3.18, we now define the counterpart of stable models for partial Herbrand interpretations.

**Definition 3.20.** A *partial stable* model of a TSS $P = (\Sigma, \Pi, R)$ is a partial Herbrand interpretation $\mathscr{M}$ for $P$ such that $\mathbf{T}^3_{P^*}(\mathscr{M}) \subseteq \mathscr{M}$, where $\subseteq$ is the information ordering; i.e., a partial stable model is a partial Herbrand interpretation closed under $\mathbf{T}^3_{P^*}$. There exists a unique least partial stable model for any TSS. The least partial stable model is consistent for any TSS. We write $\mathscr{M}^3_P$ for the least partial stable model of $P$.

**Example 3.21.** Consider the simple TSS $P$ from Example 3.19. Obviously, the partial Herbrand interpretation $(\{f^{2n}(c) \xrightarrow{a} c \mid n \geqslant 0\}, \{f^{2n+1}(c) \xrightarrow{a} c \mid n \geqslant 0\})$ is also the least partial Herbrand interpretation closed under $\mathbf{T}^3_{P^*}$. Hence, it is the least partial stable model of $P$.

Because the immediate consequence operator for partial Herbrand interpretations yields consequences on the basis of what certainly holds and what certainly does not hold, this operator also prevents unsound transition rules from being unnoticed. Unsound transition rules lead to the partial stable model $(\emptyset, \emptyset)$. Least partial stable models were shown in Ref. [32] to coincide with the *well-founded* models introduced earlier in Ref. [19] in the context of logic programming.

There exists a least partial stable model for every TSS. However, we are only interested in those TSSs of which the least partial stable model can be identified with a stable model.

**Definition 3.22.** A TSS $P$ is *meaningful* if its least partial stable model $\mathscr{M}^3_P$ is total.

In Ref. [21], a lot of evidence is given for the claim that the definition of meaningful TSS given above is the most general one without undesirable properties. Occasionally, we may also be interested in TSSs of which the least partial stable model is not total. However, excluded are occasions on which it is essential that it can be decided for every transition whether it holds in the intended model or not. Hence, excluded are occasions on which it must be derivable from the TSS concerned whether two terms are bisimilar, such as in case of a transition rule format guaranteeing that bisimulation is a congruence. Devising a *stratification* [38] for a TSS is a proven technique to check whether it is meaningful.

**Definition 3.23.** The *intended* Herbrand model of a meaningful TSS $P = (\Sigma, \Pi, R)$, written $\mathscr{M}_P$, is the Herbrand model of $P$ such that $\mathscr{M}^3_P = (\mathscr{M}_P, \mathscr{CF}^+_{\Sigma,\Pi} \setminus \mathscr{M}_P)$. $\mathscr{M}_P$ is also called the Herbrand model *associated with* $P$.

Clearly for positive TSSs, Definition 3.11 coincides with Definition 3.23. In other words, for positive TSSs, least supported models and least stable models coincide.

**Example 3.24.** In Example 3.8, a TSS was given which has two minimal supported models. The intended model of that TSS is $\{c \xrightarrow{b} c\}$. In Example 3.15, a TSS was given which has a stable model, although it has a transition rule with a premise contradicting the conclusion. That TSS does not have an intended model. In Example 3.19, a TSS was given which has a least partial stable model that is total (see also Example 3.21). The intended model of that TSS is $\{f^{2n}(c) \xrightarrow{a} c \mid n \geqslant 0\}$.

## 4. Bisimulation as a congruence

In this section, we first generalize the notion of bisimulation to TSSs that define transition relations on binding terms. Next, we generalize the transition rule format known as the panth format [38] accordingly. The main result of this paper is that the generalized panth format guarantees that generalized bisimulation is a congruence. The proof of the congruence theorem is outlined in Appendix B.

Bisimulation is a frequently used equivalence to abstract from irrelevant details of operational semantics. Originally introduced in modal logic, it was introduced in process theory in Ref. [31]. The first format guaranteeing that bisimulation is a congruence appears to be the *de Simone* format [35]. The original panth format generalizes the *ntyft/ntyxt* format of Ref. [22] for unary predicates, which in turn extends the *tyft/tyxt* format of Ref. [24] with negative premises. The original panth format also extends the *path* format of Ref. [6] with negative premises, and the path format in turn generalizes the tyft/tyxt format for unary predicates. The well-known GSOS format [9], which supports only binary predicates and both positive and negative premises, but which is more restrictive than the ntyft/ntyxt format, guarantees other useful properties.

### 4.1. Bisimulation

In Definition 4.1, we define the notion of bisimulation based on a TSS for TSSs that define transition relations on binding terms. Rule 1 is needed because we identify binding terms that can be obtained from each other by change of bound variables. Rule 2 and 3 are familiar from ordinary (strong) bisimulation. Rule 4 is reminiscent of the closure-under-substitutions property of open bisimulation equivalence, an equivalence proposed for the $\pi$-calculus in Ref. [34].

**Definition 4.1.** Let $P = (\Sigma, \Pi, R)$, where $\Sigma = (S, O)$, be a TSS. Then a *bisimulation* $B$ based on $P$ is a family of symmetric binary relations $\langle B_{\mathfrak{s}} \subseteq \mathscr{CT}_{\Sigma_{\mathfrak{s}}} \times \mathscr{CT}_{\Sigma_{\mathfrak{s}}} \rangle_{\mathfrak{s} \in \mathscr{B}(S)}$ such that, writing $B(t, t')$ for $B_{\mathfrak{s}(t)}(t, t')$:

1. $t \mathrel{\dot{\approx}} t' \Rightarrow B(t, t')$;
2. $B(t_1, t_1')$ and $p(t_1, t_2) \in \mathscr{M}_P \Rightarrow \exists t_2' \bullet p(t_1', t_2') \in \mathscr{M}_P$ and $B(t_2, t_2')$;
3. $B(t_1, t_1')$ and $p(t_1) \in \mathscr{M}_P \Rightarrow p(t_1') \in \mathscr{M}_P$;
4. $B(x_1, \ldots, x_n \,.\, t, y_1, \ldots, y_n \,.\, t') \Rightarrow \forall t_1 \in \mathscr{CT}_{\Sigma_{\mathfrak{s}(x_1)}}, \ldots, t_n \in \mathscr{CT}_{\Sigma_{\mathfrak{s}(x_n)}} \bullet$
   $B(t[t_1, \ldots, t_n/x_1, \ldots, x_n], t'[t_1, \ldots, t_n/y_1, \ldots, y_n])$.

Two closed terms $t, t' \in \mathscr{CT}_{\Sigma_{\mathfrak{s}}}$ ($\mathfrak{s} \in \mathscr{B}(S)$) are *bisimilar* in $P$, written $t \leftrightarrows_P t'$, if there exists a bisimulation $B$ such that $B(t, t')$.

**Example 4.2.** Consider the TSS for a fragment of CCS from Example 2.24. It follows from rule 2 of Definition 4.1 that $at$ and $at + at$ are bisimilar for all closed terms $t$ of this fragment. It follows from this result and rule 4 of Definition 4.1, that also $x \,.\, ax$ and $y \,.\, (ay + ay)$ are bisimilar.

### 4.2. Panth format

In Definition 4.3, we generalize the panth format of Ref. [38] to TSSs that define transition relations on binding terms. There is no essential difference between the panth format defined in Ref. [38] and the one defined here if only conventional terms are used in the transition rules. Rules 1, 3 and 4 of Definition 4.3 correspond closely to the rules of the definition given in Ref. [1] for TSSs that define transition relations on conventional terms. However, that definition would restrict the possible forms of the first argument of a conclusion to $x$ and $o(x_1, \ldots, x_n)$. Rule 2 is only needed because we chose to treat expressions of the form $\neg p(t_1)$ as abbreviations if $p$ is a binary predicate.

**Definition 4.3.** Let $P = (\Sigma, \Pi, R)$ be a TSS. Then a transition rule $r = \frac{\Phi}{\psi} \in R$ is in *panth format* if it satisfies the following restrictions:

1. for each positive premise of the form $p(t_1, t_2) \in \Phi$, the second argument $t_2$ is a variable;
2. for each negative premise of the form $\neg p(t_1, t_2) \in \Phi$, the second argument $t_2$ is a closed term;
3. the conclusion $\psi$ has the form $p(t_1)$ or $p(t_1, t_2)$, where in either case the first argument $t_1$ has one of the following forms:
   (a) $x$,
   (b) $x(x_1, \ldots, x_n)$,
   (c) $o(u_1, \ldots, u_n)$, where $u_i$ $(1 \leqslant i \leqslant n)$ is a variable or a term of the form $x_1, \ldots, x_m \,.\, x(x_1, \ldots, x_m)$;
4. the variables that occur as second argument of positive premises of the form $p(t_1, t_2)$ or as free variable of the first argument of the conclusion are mutually distinct.

The TSS $P$ is in *panth format* if each transition rule $r \in R$ is in panth format.

**Example 4.4.** Consider the TSS for a fragment of CCS from Example 2.24. It is straightforward to see that this TSS is in panth format. In all transition rules, all premises are positive and have two arguments of which the second one is a variable. Hence, restrictions 1 and 2 are met by all transition rules. In all transition rules, except the one for the recursion operator, the first argument of the conclusion has the form $o(x_1, \ldots, x_n)$. In the transition rule for the recursion operator, the first argument of the conclusion has the form $o(x_1 \,.\, x(x_1))$. Hence, restriction 3 is met by all transition rules as well. It is also easy to check that restriction 4 is met by all transition rules.

In Definitions 4.5 and 4.6, we define the notion of well-founded transition rule and the related notion of pure transition rule. These notions are used in the proofs of the congruence theorems from Refs. [10,38]. The proofs concerned make use of the result from Ref. [16] that a TSS in ntyft/ntyxt format or panth format can be transformed into an equivalent TSS in the format concerned with transition rules that are well-founded. This result extends to the generalized panth format presented above if only ordinary transition predicates, i.e., transition predicates that do not bind any variable in their arguments, are used in positive premises.

**Definition 4.5.** Let $P = (\Sigma, \Pi, R)$ be a TSS. In addition let $\Phi \subseteq \mathscr{T}^+_{\Sigma,\Pi}$. Then the *variable dependency graph* of $\Phi$ is a directed unlabeled graph with the variables occurring in $\Phi$ as nodes and as edges:

$$\{(x,x') \mid \exists p(t,t') \in \Phi \bullet x \text{ is a free variable of } t \text{ and } x' \text{ is a free variable of } t'\}.$$

$\Phi$ is *well-founded* if every backward chain of edges in its variable dependency graph is finite. A transition rule $r = \frac{\Phi}{\psi} \in R$ is *well-founded* if the set $\{\phi \in \Phi \mid \phi \in \mathscr{T}^+_{\Sigma,\Pi}\}$ is well-founded. The TSS $P$ is *well-founded* if each transition rule $r \in R$ is well-founded.

**Definition 4.6.** Let $P = (\Sigma, \Pi, R)$ be a TSS and $r \in R$. A variable $x$ is *free* in $r$ if it is a free variable of an argument of either a premise of $r$ or the conclusion of $r$, but does not occur as second argument of a positive premise of the form $p(t_1, t_2)$ or as free variable of the first argument of the conclusion. The transition rule $r$ is *pure* if $r$ is well-founded and there are no variables free in $r$.

Notice that free variables of an argument of a premise or the conclusion are not necessarily free in the rule.

The following result shows that a meaningful TSS in panth format is equivalent to one that is well-founded if only ordinary transition predicates are used in positive premises.

**Theorem 4.7.** *Let $P = (\Sigma, \Pi, R)$ be a meaningful TSS in panth format. If all transition predicates occurring in positive premises of rules in $R$ are ordinary, then there exists a well-founded TSS $P' = (\Sigma, \Pi, R')$ in panth format with $\mathscr{M}_{P'} = \mathscr{M}_P$.*

**Proof.** The proof is outlined in Appendix B. First, we prove the case of TSSs that only define binary relations. The proof amounts to careful checking of the proof of Theorem 5.4 from Ref. [16] – a well-foundedness theorem for the ntyft/ntyxt format – and adapting it to the case with variable binding operators where needed. Next, we make use of an immediate corollary of the proof of Theorem 4.9: a TSS that defines unary relations as well can be transformed into a TSS that only defines binary relations such that the models associated with the TSSs are isomorphic. □

It is an open question whether Theorem 4.7 can be proved without the restriction to ordinary transition predicates in positive premises. The course pursued in the proof

of Theorem 5.4 from Ref. [16] cannot be pursued if this restriction is not made. The problem experienced in that case is illustrated in Example 4.8.

**Example 4.8.** Consider the one-sorted signature with a sort $\mathsf{S}$, a constant $c :\rightarrow \mathsf{S}$ and a binary operator $f : \mathsf{S} \times \mathsf{S} \rightarrow \mathsf{S}$, and the following two transition predicates on terms over this signature: $\stackrel{a}{\rightarrow} : \mathsf{S} \times \mathsf{S}$ and $\stackrel{b}{\rightarrow} : \mathsf{S} \times \mathsf{S}$ . $\mathsf{S}$. The transition rule over this signature and these predicates given below is in panth format, but it is not well-founded.

$$\frac{z(x) \stackrel{a}{\rightarrow} y, y \stackrel{b}{\rightarrow} z}{f(x,y) \stackrel{a}{\rightarrow} c}$$

For a TSS containing this transition rule, we cannot conceive a transformation to an equivalent TSS that is well-founded.

### *4.3. Congruence theorem*

The following result shows that bisimulation is a congruence if well-founded transition rules in panth format are used.

**Theorem 4.9.** *Let $P = (\Sigma, \Pi, R)$ be a meaningful TSS in panth format. If $P$ is well-founded, then $\leftrightarrows_P$ is a congruence.*

**Proof.** The proof is outlined in Appendix B. First, we prove the case of TSSs that only define binary relations. The proof amounts to careful checking of the proof of Theorem 8.13 from Ref. [10] – the congruence theorem for the ntyft/ntyxt format – and adapting it to the case with variable binding operators where needed. Next, we show that each TSS that defines unary relations as well can be reduced to a TSS that only defines binary relations, while preserving bisimilarity. This is straightforward in the many-sorted case. □

Usually, proofs of congruence theorems are intricate. This includes the proof of the congruence theorem from Ref. [10]. Fortunately, the proof of Theorem 4.9 goes for the greater part exactly like the proof of that congruence theorem. The following is a corollary of Theorems 4.7 and 4.9.

**Corollary 4.10.** *Let $P = (\Sigma, \Pi, R)$ be a meaningful TSS in panth format. If all transition predicates occurring in positive premises of rules in $R$ are ordinary, then $\leftrightarrows_P$ is a congruence.*

In case not all transition predicates occurring in positive premises are ordinary, well-foundedness has to be checked.

**Example 4.11.** Consider the TSS for a fragment of CCS from Example 2.24. Because it is a positive TSS, we have immediately that it is a meaningful TSS. Moreover, it is in panth format (see also Example 4.4). All transition predicates are ordinary. Hence, by Corollary 4.10, bisimilation is a congruence in that TSS.

## 5. Given sorts and parametrization

In various applications of TSSs, it is impractical and unnecessary to provide the terms of certain sorts with an operational semantics because there exists a fully established semantics for them. We will call such sorts *given sorts*. It is common to identify terms of given sorts if they are semantically equivalent. This can be formalized as follows. First of all, we introduce $\approx\!\!\!\!\approx$, the (sort-indexed) family of least congruence relations on binding terms that includes both $\approx$ and the equivalence induced by the semantics for the terms of given sorts. Next, we replace in the definitions of Sections 3 and 4 all occurrences of $\approx$ by $\approx\!\!\!\!\approx$. These replacements slightly alter the notions of transition system, Herbrand interpretation, proof, partial Herbrand interpretation, and bisimulation. Indirectly, they also alter the immediate consequence operators ($\mathbf{T}_P$ and $\mathbf{T}_P^3$), the partial stable models for a TSS, and the intended Herbrand model of a TSS ($\mathscr{M}_P$). In all cases, the alteration is simply that more terms are identified. Finally, we replace clause (c) of rule 3 in the definition of the panth format (Definition 4.3) by the following clause:

(c′) $o(u_1,\ldots,u_n)$, where $u_i$ $(1 \leqslant i \leqslant n)$ is a variable or a term of the form $x_1,\ldots,x_m \,.\, x(x_1,\ldots,x_m)$ or a term of a given sort;

This modification permits, for given sorts $s$, that a term of sort $s$ is used where the original rule only permits that a variable of sort $s$ is used. The congruence theorem goes through in the case of TSSs with given sorts (obviously with the altered definitions of bisimulation and panth format). The proof goes like the proof of Theorem 4.9.

Distinguishing given sorts does not only make it possible to relax the panth format. It also allows for TSSs that define parametrized transition relations, where the parameters are closed terms of given sorts. Put differently, we can relax the restriction that a transition predicate $p$ is a predicate $p : \mathfrak{s}_1 \times \cdots \times \mathfrak{s}_n$ with $1 \leqslant n \leqslant 2$ to the restriction that a transition predicate $p$ is a predicate $p : \mathfrak{s}_1 \times \cdots \times \mathfrak{s}_n$ with at most two sorts among $\mathfrak{s}_1,\ldots,\mathfrak{s}_n$ that are not given sorts. Suppose that $p$ is a parametrized transition predicate $p : \mathfrak{s}_1 \times \cdots \times \mathfrak{s}_n$ and $i_1,\ldots,i_k$ $(n-2 \leqslant k \leqslant n-1)$ are the indices of the given sorts in increasing order. Then we can take a fresh predicate $p_{t_1,\ldots,t_k}$ for each closed term $t_1$ of sort $\mathfrak{s}_{i_1}$, …, closed term $t_k$ of sort $\mathfrak{s}_{i_k}$. It is easy to see that carrying on in this way, we can reduce any TSS that defines parametrized transition relations to a TSS that only defines unparametrized – unary and binary – transition relations, while preserving bisimilarity.

**Example 5.1.** In Appendix A, a TSS of $\mathrm{BPA}^{\mathrm{sat}}$ with integration (see Ref. [5]) is given. It is a TSS that defines parametrized transition relations. The sort $\mathbb{R}_{\geqslant 0}$ of non-negative real numbers and the sort $\mathscr{P}(\mathbb{R}_{\geqslant 0})$ of sets of non-negative real numbers are considered given sorts. There is one transition rule with a negative premise. Moreover, a variable binding operator, viz., the integration operator $\int$, is involved. There exists a stratification for this TSS and consequently it is meaningful. It is easy to see that the TSS is in the relaxed panth format. In addition, all transition predicates concerned are ordinary. Hence, $\leftrightarrows_P$ is a congruence.

In Ref. [18], in which the tyft/tyxt format has essentially been extended to deal with many-sortedness, given sorts and parametrized transition relations, there is a

need for restrictions on the rule format concerning the distinctness of the free variables of the arguments of given sorts and a compatibility condition with respect to the equivalence induced by the semantics for the terms of given sorts. We do not need such restrictions because we look at transition systems that consist of transition relations on $\leftrightarroweq$-equivalence classes of closed terms (see Section 3), instead of transition relations on closed terms. This approach to the meaning of TSSs, chosen on semantic grounds, guarantees that terms of given sorts cannot prevent bisimulation equivalence from being a congruence.

**Example 5.2.** From the TSS of $\mathrm{BPA}^{\mathrm{sat}}$ given in Appendix A, we can immediately prove $\langle \tilde{a}, 0\rangle \xrightarrow{a} \langle \surd, 0\rangle$ and $\langle \tilde{a}, 0\rangle \xrightarrow{a} \langle \surd, 1 \dot{-} 1\rangle$ (for any action $a$). In our approach, these formulas refer to the same transition. In the approach of Ref. [18], they refer to different transitions – which can be exchanged for each other in so far as bisimulation is concerned.

## 6. Concluding remarks

The notion of TSS was first introduced in Ref. [24] and then generalized in Ref. [6,10,22,38]. We generalized it further to cover variable binding operators and many-sortedness. We found that the notions of bisimulation and panth format generalize naturally to the generalized TSSs, and moreover that in the generalized setting bisimulation is still a congruence for meaningful TSSs in panth format. Therefore, we expect that the applicability of TSSs to provide process calculi, specification languages and programming languages with an operational semantics has been improved. The generalized TSSs can amongst other things deal with: the integration operator $\int$ of real time ACP [4], the sum operator $\sum$ of $\mu$CRL [23], and the recursion operator $\mu$ of CSP [25] and CCS [28]. If the $\pi$-calculus [29] would separate names and variables, guaranteeing that distinct constants remain distinct, $\pi$-calculus features such as input action prefixes $x(y)$ and the restriction operator $\nu$ could be dealt with as well.

Our main motivation to take up the work presented in this paper stems directly from work on an integrated treatment of all versions of ACP with timing [5]. That work created the need of a generalization of the framework from Ref. [38] that takes variable binding operators and many-sortedness into account. The notion of TSS was already generalized to deal with variable binding operators and many-sortedness in Ref. [17]. That paper gives syntactic criteria for the (operational) conservativity property of extensions, but not for the congruence property of bisimulation equivalence. Initially, we tried to generalize the congruence result of Ref. [38] to the generalized TSSs of Ref. [17]. This turned out to be far from straightforward because of the distinction made between formal and actual variables, formal and actual terms, formal and actual substitutions, etc. In addition, it was an obstacle that there is little semantic clarification in Ref. [17] of the notations introduced. Therefore, we chose to introduce an alternative generalization. We did not give syntactic criteria for conservativity of extensions, which is important in cases where an existing calculus or language is extended with new features. We keep this topic for future research, but we expect that it is easy to generalize the relevant results of Ref. [12] or to adapt the results of Ref. [17].

The generalized TSSs presented in this paper are TSSs that define transition relations on binding terms. Binding terms are basically second-order terms of a restricted kind, suitable to deal with variable binding operators. As a result, binding terms are not meant to deal with general second-order operators. Binding terms do not support higher-order operators other than the second-order operators that can be regarded as variable binding operators. Consequently, the generalized TSSs are not appropriate to provide higher-order process calculi and higher-order functional programming languages with an operational semantics. Approaches to structural operational semantics for the higher-order case has been studied extensively by others.

Approaches for higher-order process calculi are investigated, for example, in Ref. [8]. In that paper attention is concentrated on CHOCS-like process calculi [37]. The paper introduces higher-order transition rules that define typed transition relations. Higher-order versions of bisimulation and the GSOS format are also proposed. Interesting is that for the restriction to the second-order case another version of bisimulation, called white bisimulation, is proposed and a corresponding congruence result is given. White bisimulation generalizes bisimulation as defined in Section 4.1 from binding terms to general second-order terms.

Approaches to structural operational semantics for (higher-order) functional programming languages are investigated, for example, in Ref. [26]. Just like most other papers on this issue, the focus in that paper is a restricted kind of transition systems and a variant of bisimulation, known as evaluation systems and applicative bisimulation, respectively. The proposed approach requires to distinguish two kinds of variables, terms, substitutions, etc., – like in Ref. [17]. A transition rule format is proposed for which a congruence result is given. The format concerned is virtually incomparable with the generalized panth format defined in Section 4.2. In Ref. [33], which has been inspired by Howe [26], another format, called the GDSOS format, is introduced. That format is more restrictive, but guarantees other properties which permit, for example, to use a kind of fixed-point induction.

Higher-order features of process calculi, and programming languages and specification languages, can sometimes be dealt with in a first-order framework by apposite choice of auxiliary sorts and operators, and parametrized transition predicates in the sense of Section 5. This is illustrated in Ref. [7]. In that paper, a generalization of the tyft/tyxt format, called the promoted tyft/tyxt format, is proposed for which a congruence result is given. This format applies to parametrized transition predicates with only one parameter. The promoted tyft/tyxt format impose restrictions on the parameter which are weaker than the requirement that the sort of the parameter must be a given sort.

## Appendix A. TSS of $BPA^{sat}$ with integration

In this appendix, we give the signature, the domain of transition predicates and the transition rules of $BPA^{sat}$ with integration. In $BPA^{sat}$, basic standard real time process algebra with absolute timing, parallelism and communication are not considered. The integration operator $\int$ provides for alternative composition over a

continuum of alternatives. In $\mathrm{BPA}^{\mathrm{sat}}$, a set $\mathsf{A}$ of actions is assumed. $\mathsf{P}$ is the sort of processes.

The signature of $\mathrm{BPA}^{\mathrm{sat}}$ consists of an *urgent action* constant $\tilde{a} :\rightarrow \mathsf{P}$ for each $a \in \mathsf{A}$, the *urgent deadlock* constant $\tilde{\delta} :\rightarrow \mathsf{P}$, the *immediate deadlock* constant $\dot{\delta} :\rightarrow \mathsf{P}$, the *alternative composition* operator $+ : \mathsf{P} \times \mathsf{P} \to \mathsf{P}$, the *sequential composition* operator $\cdot :\rightarrow \mathsf{P} \times \mathsf{P} \to \mathsf{P}$, the *absolute delay* operator $\sigma_{\mathsf{abs}} : \mathbb{R}_{\geqslant 0} \times \mathsf{P} \to \mathsf{P}$, and the *integration* (variable binding) operator $\int : \mathscr{P}(\mathbb{R}_{\geqslant 0}) \times \mathbb{R}_{\geqslant 0}.\mathsf{P} \to \mathsf{P}$.

The process $\tilde{a}$ is only capable of performing action $a$, immediately followed by successful termination, at time 0. The process $\tilde{\delta}$, although existing at time 0, is incapable of doing anything. The process $\dot{\delta}$ is the process that exhibits inconsistent timing at time 0. This means that $\dot{\delta}$, different from $\tilde{\delta}$, does not exist at time 0 and hence causes a time stop at time 0. The process $\sigma^p_{\mathsf{abs}}(x)$ is the process $x$ shifted in time by $p$. Thus, the process $\sigma^p_{\mathsf{abs}}(\tilde{\delta})$ is capable of idling from time 0 up to and including time $p$ – and at time $p$ it gets incapable of doing anything – whereas the process $\sigma^p_{\mathsf{abs}}(\dot{\delta})$ is only capable of idling from time 0 upto, but not including, time $p$. So $\sigma^p_{\mathsf{abs}}(\dot{\delta})$ cannot reach time $p$.

We need four kinds of transition predicates:

| | | |
|---|---|---|
| a predicate $\langle \_, \_\rangle \xrightarrow{a} \langle \_, \_\rangle$ | $: \mathsf{P} \times \mathbb{R}_{\geqslant 0} \times \mathsf{P} \times \mathbb{R}_{\geqslant 0}$ | for each $a \in \mathsf{A}$, |
| a predicate $\langle \_, \_\rangle \xrightarrow{a} \langle \surd, \_\rangle$ | $: \mathsf{P} \times \mathbb{R}_{\geqslant 0} \times \mathbb{R}_{\geqslant 0}$ | for each $a \in \mathsf{A}$, |
| a predicate $\langle \_, \_\rangle \mapsto \langle \_, \_\rangle$ | $: \mathsf{P} \times \mathbb{R}_{\geqslant 0} \times \mathsf{P} \times \mathbb{R}_{\geqslant 0}$, | |
| a predicate $\mathsf{ID}(\_)$ | $: \mathsf{P}$. | |

The four kinds of transition predicates are called the *action step*, *action termination*, *time step* and *immediate deadlock* predicates. We will only define transition relations for which $\langle t, p\rangle \xrightarrow{a} \langle t', q\rangle$ and $\langle t, p\rangle \xrightarrow{a} \langle \surd, q\rangle$ never hold if $p \neq q$. The four kinds of transition predicates can be explained as follows:

| | |
|---|---|
| $\langle t, p\rangle \xrightarrow{a} \langle t', p\rangle$: | process $t$ is capable of first performing action $a$ at time $p$ and then proceeding as process $t'$; |
| $\langle t, p\rangle \xrightarrow{a} \langle \surd, p\rangle$: | process $t$ is capable of first performing action $a$ at time $p$ and then terminating successfully; |
| $\langle t, p\rangle \mapsto \langle t', q\rangle$: | process $t$ is capable of first idling from time $p$ to time $q$ and then proceeding as process $t'$; |
| $\mathsf{ID}(t)$: | process $t$ is not capable of reaching time 0. |

The transition rules for $\mathrm{BPA}^{\mathrm{sat}}$ with integration are given in Tables 1 and 2. These rules are easy to understand. We will only explain the rules for the absolute delay operator ($\sigma_{\mathsf{abs}}$). The rules in the second row express that the action related capabilities of a process $\sigma^0_{\mathsf{abs}}(x)$ at time $p$ include those of process $x$ at time $p$ and that the action related capabilities of a process $\sigma^r_{\mathsf{abs}}(x)$ $(r > 0)$ at time $p + r$ include those of process $x$ at time $p$ shifted in time by $r$. The first and second rule in the fifth row express that a process $\sigma^r_{\mathsf{abs}}(x)$ $(r > 0)$ can idle from any time $p \geqslant 0$ to any time $p' < r$ and that it can also idle to time $r$ provided that process $x$ can reach time 0. The third rule in the fifth row expresses that the time related capabilities of a process $\sigma^q_{\mathsf{abs}}(x)$ $(q \geqslant 0)$ at time $p + q$ include those of process $x$ at time $p$ shifted in time by $q$.

Table 1
Rules for operational semantics of $\mathrm{BPA}^{\mathrm{sat}}$

$$\frac{}{\langle \tilde{a}, 0\rangle \xrightarrow{a} \langle \surd, 0\rangle}$$

$$\frac{\langle x,p\rangle \xrightarrow{a} \langle x',p\rangle}{\langle \sigma^0_{\mathsf{abs}}(x),p\rangle \xrightarrow{a} \langle x',p\rangle} \quad \frac{\langle x,p\rangle \xrightarrow{a} \langle x',p\rangle, 0<r}{\langle \sigma^r_{\mathsf{abs}}(x),p+r\rangle \xrightarrow{a} \langle \sigma^r_{\mathsf{abs}}(x'),p+r\rangle} \quad \frac{\langle x,p\rangle \xrightarrow{a} \langle \surd,p\rangle}{\langle \sigma^q_{\mathsf{abs}}(x),p+q\rangle \xrightarrow{a} \langle \surd,p+q\rangle}$$

$$\frac{\langle x,p\rangle \xrightarrow{a} \langle x',p\rangle}{\langle x+y,p\rangle \xrightarrow{a} \langle x',p\rangle} \quad \frac{\langle y,p\rangle \xrightarrow{a} \langle y',p\rangle}{\langle x+y,p\rangle \xrightarrow{a} \langle y',p\rangle} \quad \frac{\langle x,p\rangle \xrightarrow{a} \langle \surd,p\rangle}{\langle x+y,p\rangle \xrightarrow{a} \langle \surd,p\rangle} \quad \frac{\langle y,p\rangle \xrightarrow{a} \langle \surd,p\rangle}{\langle x+y,p\rangle \xrightarrow{a} \langle \surd,p\rangle}$$

$$\frac{\langle x,p\rangle \xrightarrow{a} \langle x',p\rangle}{\langle x\cdot y,p\rangle \xrightarrow{a} \langle x'\cdot y,p\rangle} \quad \frac{\langle x,p\rangle \xrightarrow{a} \langle \surd,p\rangle}{\langle x\cdot y,p\rangle \xrightarrow{a} \langle y,p\rangle} \quad \frac{}{\mathsf{ID}(\dot{\delta})} \quad \frac{\mathsf{ID}(x)}{\mathsf{ID}(\sigma^0_{\mathsf{abs}}(x))} \quad \frac{\mathsf{ID}(x),\mathsf{ID}(y)}{\mathsf{ID}(x+y)} \quad \frac{\mathsf{ID}(x)}{\mathsf{ID}(x\cdot y)}$$

$$\frac{p<p',p'<r}{\langle \sigma^r_{\mathsf{abs}}(x),p\rangle \mapsto \langle \sigma^r_{\mathsf{abs}}(x),p'\rangle} \quad \frac{\neg\mathsf{ID}(x),p<r}{\langle \sigma^r_{\mathsf{abs}}(x),p\rangle \mapsto \langle \sigma^r_{\mathsf{abs}}(x),r\rangle} \quad \frac{\langle x,p\rangle \mapsto \langle x,p'\rangle}{\langle \sigma^q_{\mathsf{abs}}(x),p+q\rangle \mapsto \langle \sigma^q_{\mathsf{abs}}(x),p'+q\rangle}$$

$$\frac{\langle x,p\rangle \mapsto \langle x,p'\rangle}{\langle x+y,p\rangle \mapsto \langle x+y,p'\rangle} \quad \frac{\langle y,p\rangle \mapsto \langle y,p'\rangle}{\langle x+y,p\rangle \mapsto \langle x+y,p'\rangle} \quad \frac{\langle x,p\rangle \mapsto \langle x,p'\rangle}{\langle x\cdot y,p\rangle \mapsto \langle x\cdot y,p'\rangle}$$

$(a \in \mathsf{A},\ x,x',y,y' : \mathsf{P},\ p,p',q,r,v : \mathbb{R}_{\geqslant 0})$

Table 2
Additional rules for integration

$$\frac{\langle P(q),p\rangle \xrightarrow{a} \langle x',p\rangle, q\in V}{\langle \int_{v\in V} P(v),p\rangle \xrightarrow{a} \langle x',p\rangle} \quad \frac{\langle P(q),p\rangle \xrightarrow{a} \langle \surd,p\rangle, q\in V}{\langle \int_{v\in V} P(v),p\rangle \xrightarrow{a} \langle \surd,p\rangle} \quad \frac{\langle P(q),p\rangle \mapsto \langle x',p'\rangle, q\in V}{\langle \int_{v\in V} P(v),p\rangle \mapsto \langle \int_{v\in V} P(v),p'\rangle}$$

$$\frac{}{\mathsf{ID}(\int_{v\in\emptyset} P(v))} \quad \frac{\mathsf{ID}(P(q))}{\mathsf{ID}(\int_{v\in\{q\}} P(v))} \quad \frac{\mathsf{ID}(\int_{v\in V} P(v)),\mathsf{ID}(\int_{v\in W} P(v))}{\mathsf{ID}(\int_{v\in V\cup W} P(v))}$$

$(a \in \mathsf{A}, P : \mathbb{R}_{\geqslant 0} \,.\, \mathsf{P},\ x' : \mathsf{P},\ p,q,r,v : \mathbb{R}_{\geqslant 0}, V, W : \mathscr{P}(\mathbb{R}_{\geqslant 0}))$

## Appendix B. Outline of proofs

**Theorem 4.7** (Well-foundedness). *Let $P = (\Sigma, \Pi, R)$ be a meaningful TSS in panth format. If all transition predicates occurring in positive premises of rules in $R$ are ordinary, then there exists a well-founded TSS $P' = (\Sigma, \Pi, R')$ in panth format with $\mathscr{M}_{P'} = \mathscr{M}_P$.*

**Proof.** If Theorem 5.4 from Ref. [16] goes through for many-sorted terms with variable binding in case all transition predicates occurring in positive premises are ordinary, the theorem follows immediately. It is straightforward to check that Theorem

5.4 from Ref. [16] goes through. The proof presented in Ref. [16] makes, directly or indirectly, use of Lemma 3.2 and adaptations of Lemmas and Theorems 4.1, 4.5 and 4.9 from that paper. It is immediately clear that Lemma 3.2 goes through for many-sorted terms with variable binding seeing that its proof does not depend on the term structure. The proof of the adaptation of Lemma 4.1 needs a slight adaptation of the substitutions because the operators may now be variable binding operators. In the proofs of the adaptations of Theorems 4.5 and 4.9, it is now necessary to make case distinctions wherever the form of the first argument of premises or conclusions matters because it may have the form $x(x_1, \ldots, x_n)$ as well. However, the additional cases are similar to the the original ones. □

**Theorem 4.9** (Congruence). *Let $P = (\Sigma, \Pi, R)$, where $\Sigma = (S, O)$, be a meaningful TSS in panth format. If $P$ is well-founded, then $\leftrightarrows_P$ is a congruence.*

**Proof.** First, we prove the case that all predicates in $\Pi$ are binary predicates. Next, we show that, if there are also unary predicates in $\Pi$, we can construct from the TSS $P$ a new TSS $P'$ without unary predicates such that two closed terms over $\Sigma$ are bisimilar in $P$ if and only if they are bisimilar in $P'$.

Theorem 8.13 from Ref. [10] covers the case that all predicates in $\Pi$ are binary predicates for single-sorted terms without variable binding. In that theorem the requirement of well-foundedness is omitted because it follows from Theorem 5.4 from Ref. [16] that it can be omitted. Theorem 8.13 from Ref. [10] goes through for many-sorted terms with variable binding if we add the requirement of well-foundedness. The proof presented in Ref. [10] makes, directly or indirectly, use of the Lemmas, Theorems and Corollaries 5.5–5.8, 8.6, 8.8, 8.9 and 8.12 from that paper as well as Theorem 5.4 from Ref. [16]. We do not have to check that Theorem 5.4 from Ref. [16] goes through because we do not omit the requirement of well-foundedness in our congruence theorem. It is immediately clear that Lemmas, Theorems and Corollaries 5.5–5.8 go through for many-sorted terms with variable binding seeing that their proofs do not depend on the term structure. It requires little effort to check that Lemmas 8.6, 8.8 and 8.12 go through as well. The proof of Lemma 8.6 needs a slight adaptation of the substitutions $\sigma_f$ because $f$ may now be a variable binding operator. The proof of Lemma 8.8 remains straightforward after the adaptation of $R_P$, meant to be the minimal congruence that includes $\leftrightarrows_P$, to terms in which variables are bound. The proof of Lemma 8.12 simply combines Corollary 5.8 and Lemma 8.9.

Surprisingly, it is also straightforward to check that Lemma 8.9, which is the main lemma, goes through. That lemma consists of two parts. In the proof of both parts, it is now necessary to show as well that

$$x_1, \ldots, x_n \,.\, u \,R_P\, y_1, \ldots, y_n \,.\, v \Rightarrow \forall t_1 \in \mathscr{CT}_{\Sigma_{\mathfrak{s}(x_1)}}, \ldots, t_n \in \mathscr{CT}_{\Sigma_{\mathfrak{s}(x_n)}} \bullet$$
$$u[t_1, \ldots, t_n/x_1, \ldots, x_n] \,R_P\, v[t_1, \ldots, t_n/y_1, \ldots, y_n].$$

This follows immediately from the definitions of $\leftrightarrows_P$ and $R_P$. The proof of the first part makes use of Claim 8.10 and the proof of the second part makes use of Claim 8.11. Both claims concern the existence of a substitution with certain properties. In the proof of property (b) of both claims, now two cases have to be distinguished: (i) $x_i : s$ for some $s \in S$ and (ii) $x_i : \mathfrak{s}$ for some $\mathfrak{s} = s_1 \times \cdots \times s_n \,.\, s \in \mathscr{B}(S)$. The proof for case (ii) is easy using the following fact:

$$\sigma(x_i) \notin \mathscr{V}_{\mathfrak{s}} \Rightarrow \sigma(x_i) \doteq \sigma(y_1, \ldots, y_n \,.\, x_i(y_1, \ldots, y_n)).$$

This fact follows immediately from the definition of substitution. In the proof of both parts, it is also necessary to take into account that the last transition rule applied in a proof may have a conclusion of the form $p(x'(x_1, \ldots, x_n), t)$. This means that variants of Claims 8.10 and 8.11 have to be proved that include the additional property

$$(\mathrm{b}')\ \text{if } x = x' \text{ then } \sigma'(x) = x_1, \ldots, x_n \,.\, f(x_1, \ldots, x_n),$$

which is easy to prove.

Constructing from the TSS $P$ a new TSS $P'$ without unary predicates that preserves bisimilarity is for many-sorted terms much easier than for single-sorted terms (see Theorem 4.5 in Ref. [38] for an appropriate construction in the single-sorted case). For many-sorted terms with variable binding, the construction of $P' = (\Sigma', \Pi', R')$, where $\Sigma' = (S', O')$, goes as follows:

1. $S' = S \cup \{s_\xi\}$, where $s_\xi \in \mathscr{S} \setminus S$;
2. $O' = O \cup \{\xi\}$, where $\xi :\rightarrow s_\xi$;
3. $\Pi' = \{p \in \Pi \mid \exists \mathfrak{s}_1 \in S, \mathfrak{s}_2 \in \mathscr{B}(S) \bullet p : \mathfrak{s}_1 \times \mathfrak{s}_2\} \cup$
   $\{p_\xi \in \mathscr{P} \mid \exists p \in \Pi, s \in S \bullet p : s \text{ and } p_\xi : s \times s_\xi\}$;
4. $R' = \left\{ \dfrac{\Xi(\phi) \mid \phi \in \Phi}{\Xi(\psi)} \middle| \dfrac{\Phi}{\psi} \in R \right\}$,

where

$$\Xi(p(t_1, t_2)) = p(t_1, t_2), \qquad \Xi(\neg p(t_1, t_2)) = \neg p(t_1, t_2),$$
$$\Xi(p(t)) = p_\xi(t, z)\ (z \text{ fresh and } z \in \mathscr{V}_\xi), \qquad \Xi(\neg p(t)) = \neg p_\xi(t, \xi).$$

That is, $s_\xi$ is a fresh sort, $\xi$ is the only term of sort $s_\xi$, binary predicates $p_\xi$ take the place of unary predicates $p$, and the new transition rules are obtained from the old ones by replacing each positive transition formula $p(t)$ by $p_\xi(t, z)$ and each negative transition formula $\neg p(t)$ by $\neg p_\xi(t, \xi)$. The proviso that the variables $z$ used in these replacements must be fresh means that they are mutually distinct. It follows immediately that $\sigma(p_\xi(t, x)) = p_\xi(\sigma(t), \xi)$ for all closed substitutions $\sigma$. Next it is easy to see that $p(t) \in \mathscr{M}_P \iff p_\xi(t, \xi) \in \mathscr{M}_{P'}$, $p(t_1, t_2) \in \mathscr{M}_P \iff p(t_1, t_2) \in \mathscr{M}_{P'}$, and $\xi \leftrightarrows_{P'} \xi$. Consequently, for all $\mathfrak{s} \in \mathscr{B}(S)$, for all $t, t' \in \mathscr{CT}_{\Sigma_{\mathfrak{s}}}$, $t \leftrightarrows_P t' \iff t \leftrightarrows_{P'} t'$. □

## References


[1] L. Aceto, W.J. Fokkink, C. Verhoef, Structural operational semantics, in: J.A. Bergstra, A. Ponse, S.A. Smolka (Eds.), Handbook of Process Algebra, Elsevier, Amsterdam, 2001.

[2] P. Aczel, Frege structures and the notions of proposition, truth and set, in: J. Barwise, H.J. Keisler, K. Kunen (Eds.), The Kleene Symposium, North-Holland, Amsterdam, 1980, pp. 31–59.

[3] K.R. Apt, R.N. Bol, Logic programming and negation: a survey, J. Logic Programming 19–20 (1994) 9–71.

[4] J.C.M. Baeten, J.A. Bergstra, Real time process algebra, Formal Aspects Comput 3 (2) (1991) 142–188.

[5] J.C.M. Baeten, C.A. Middelburg, Process algebra with timing: real time and discrete time, in: J.A. Bergstra, A. Ponse, S.A. Smolka (Eds.), Handbook of Process Algebra, Elsevier, Amsterdam, 2001.

[6] J.C.M. Baeten, C. Verhoef, A congruence theorem for structured operational semantics with predicates, in: E. Best (Ed.), CONCUR'93, Lecture Notes in Computer Science, vol. 715, Springer, Berlin, 1993, pp. 477–492.

[7] K.L. Bernstein, A congruence theorem for structured operational semantics of higher-order languages, in: LICS '98, IEEE Computer Science Press, Silver Spring, MD, 1998, pp. 153–164.
[8] B. Bloom, CHOCOLATE: Calculi of higher order communication and lambda terms, in: Symposium on Principles of Programming Languages, ACM Press, New York, 1994, pp. 339–347.
[9] B. Bloom, S. Istrail, A.R. Meyer, Bisimulation can't be traced, J. ACM 42 (1995) 232–268.
[10] R.N. Bol, J.F. Groote, The meaning of negative premises in transition system specifications, J. ACM 43 (1996) 863–914.
[11] L. Chen, An interleaving model for real-time systems, in: A. Nerode, M. Taitslin (Eds.), Symposium on Logical Foundations of Computer Science, Lecture Notes in Computer Science, vol. 620, Springer, Berlin, 1992, pp. 81–92.
[12] P.R. D'Argenio, C. Verhoef, A conservative extension theorem in process algebras with inequalities, Theoret. Comput. Sci. 177 (1997) 351–380.
[13] M.H. van Emden, R.A. Kowalski, The semantics of predicate logic as a programming language, J. ACM 23 (1976) 733–742.
[14] M.P. Fiore, G.D. Plotkin, D. Turi, Abstract syntax and variable binding, in: LICS '99, IEEE Computer Science Press, Silver Spring, MD, 1999, pp. 199–203.
[15] M. Fitting, A Kripke-Kleene semantics for general logic programs, J. Logic Programming 2 (1985) 295–312.
[16] W.J. Fokkink, R.J. van Glabbeek, Ntyft/ntyxt rules reduce to ntree rules, Informat. Comput. 126 (1996) 1–10.
[17] W.J. Fokkink, C. Verhoef, A conservative look at operational semantics with variable binding, Informat. Comput. 146 (1998) 24–54.
[18] V.C. Galpin, Comparison of process algebra equivalences using formats, in: J. Wiedermann, P. van Emde Boas, M. Nielsen (Eds.), Proceedings 26th ICALP, Lecture Notes in Computer Science, vol. 1644, Springer, Berlin, 1999, pp. 341–350.
[19] A. van Gelder, K. Ross, J. Schlipf, The well-founded semantics for general logic programs, J. ACM 38 (1991) 620–650.
[20] M. Gelfond, V. Lifschitz, The stable model semantics for logic programming, in: R.A. Kowalski, K.A. Bowen (Eds.), Fifth International Conference and Symposium on Logic Programming, ALP, MIT Press, Cambridge, MA, 1988, pp. 1070–1080.
[21] R.J. van Glabbeek, The meaning of negative premises in transition system specifications II, in: F. Meyer auf der Heide, B. Monien (Eds.), Proceedings 23th ICALP, Lecture Notes in Computer Science, vol. 1099, Springer, Berlin, 1996, pp. 502–513.
[22] J.F. Groote, Transition system specifications with negative premises, Theoret. Comput. Sci. 118 (1993) 263–299.
[23] J.F. Groote, A. Ponse, The syntax and semantics of $\mu$CRL, in: A. Ponse, C. Verhoef, S.F.M. van Vlijmen (Eds.), Algebra of Communicating Processes 1994, Workshop in Computing Series, Springer, Berlin, 1995, pp. 26–62.
[24] J.F. Groote, F.W. Vaandrager, Structured operational semantics and bisimulation as a congruence, Informat. Comput. 100 (1992) 202–260.
[25] C.A.R. Hoare, Communicating Sequential Processes, Prentice-Hall, Englewood Cliffs, 1985.
[26] D.J. Howe, Proving congruence of bisimulation in functional programming languages, Informat. Comput. 124 (1996) 103–112.
[27] J.W. Klop, V. van Oostrom, F. van Raamsdonk, Combinatory reduction systems: introduction and survey, Theoret. Comput. Sci. 121 (1993) 279–308.
[28] R. Milner, Communication and Concurrency, Prentice-Hall, Englewood Cliffs, 1989.
[29] R. Milner, J. Parrow, D. Walker, A calculus of mobile processes: Part I and II, Informat. Comput. 100 (1992) 1–77.
[30] X. Nicollin, J. Sifakis, The algebra of timed processes ATP: theory and application, Informat. Comput. 114 (1994) 131–178.
[31] D. Park, Concurrency and automata on infinite sequences, in: P. Deussen (Ed.), Proceedings 5th GI Conference, Lecture Notes in Computer Science, vol. 104, Springer, Berlin, 1981, pp. 167–183.
[32] T.C. Przymusinski, The well-founded semantics coincides with the three-valued stable semantics, Fundamenta Informaticae 13 (1990) 445–463.
[33] D. Sands, From SOS rules to proof principles: an operational metatheory for functional languages, in: Symposium on Principles of Programming Languages, ACM Press, New York, 1997, pp. 428–441.
[34] D. Sangiorgi, A theory of bisimulation for the $\pi$-calculus, Acta Informatica 33 (1996) 69–97.

[35] R. de Simone, Higher-level synchronising devices in MEIJE–SCCS, Theoret. Comput. Sci. 37 (1985) 245–267.
[36] SunYong, An algebraic generalization of Frege structures – binding algebras, Theoret. Comput. Sci. 211 (1999) 189–232.
[37] B. Thomsen, A theory of higher order communicating systems, Informat. Comput. 116 (1995) 38–57.
[38] C. Verhoef, A congruence theorem for structured operational semantics with predicates and negative premises, Nordic J. Comput. 2 (1995) 274–302.